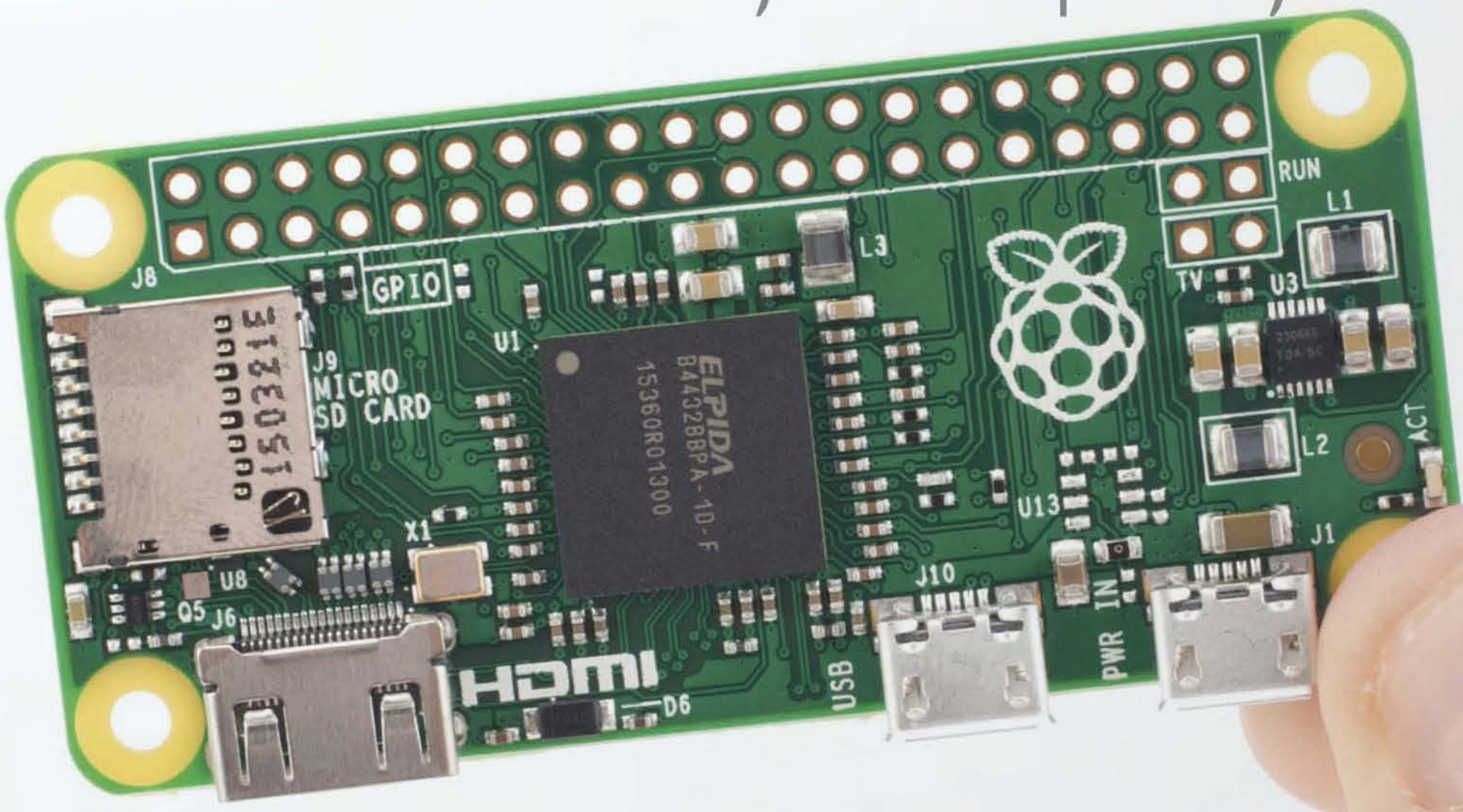
RasPi

DESIGN
BUILD
CODE

18

Get hands-on with your Raspberry Pi



50 Ways to Master

PI ZERO

PLUS Models A+, B+ and 2B

Welcome



There's a new kind of Pi invading living rooms, workshops and garages everywhere, and it costs just $5: the Raspberry Pi Zero. It's a little bit different to what you're used to and an incredibly capable little thing – swipe forward a couple of pages to find all the key facts. When you're ready, check out this month's big feature to find 50 ways that you can master the new device. The great thing is that while some of those tips and tricks are specific to the Zero, the majority will actually work with any model of Pi, from A to 2B. Also this month we've got a great project to show you that combines LEGO and the Internet of Things into a mini smart city that you can interact with over Twitter. Enjoy!

Gavin Thomas

Editor

Get inspired

Discover the RasPi community's best projects

Expert advice

Got a question? Get in touch and we'll give you a hand

Easy-to-follow guides

Learn to make and code gadgets with Raspberry Pi



From the makers of

LinuxUser & Developer

Join the conversation at...

@linuxusermag | Linux User & Developer | RasPi@imagine-publishing.co.uk



Contents

What is the Raspberry Pi Zero?
All the key facts in a handy FAQ

50 Ways to Master Your Pi
Top tips for your Zero, A+, B+ and 2B

Internet of LEGO
Find out how to bring a LEGO city to life

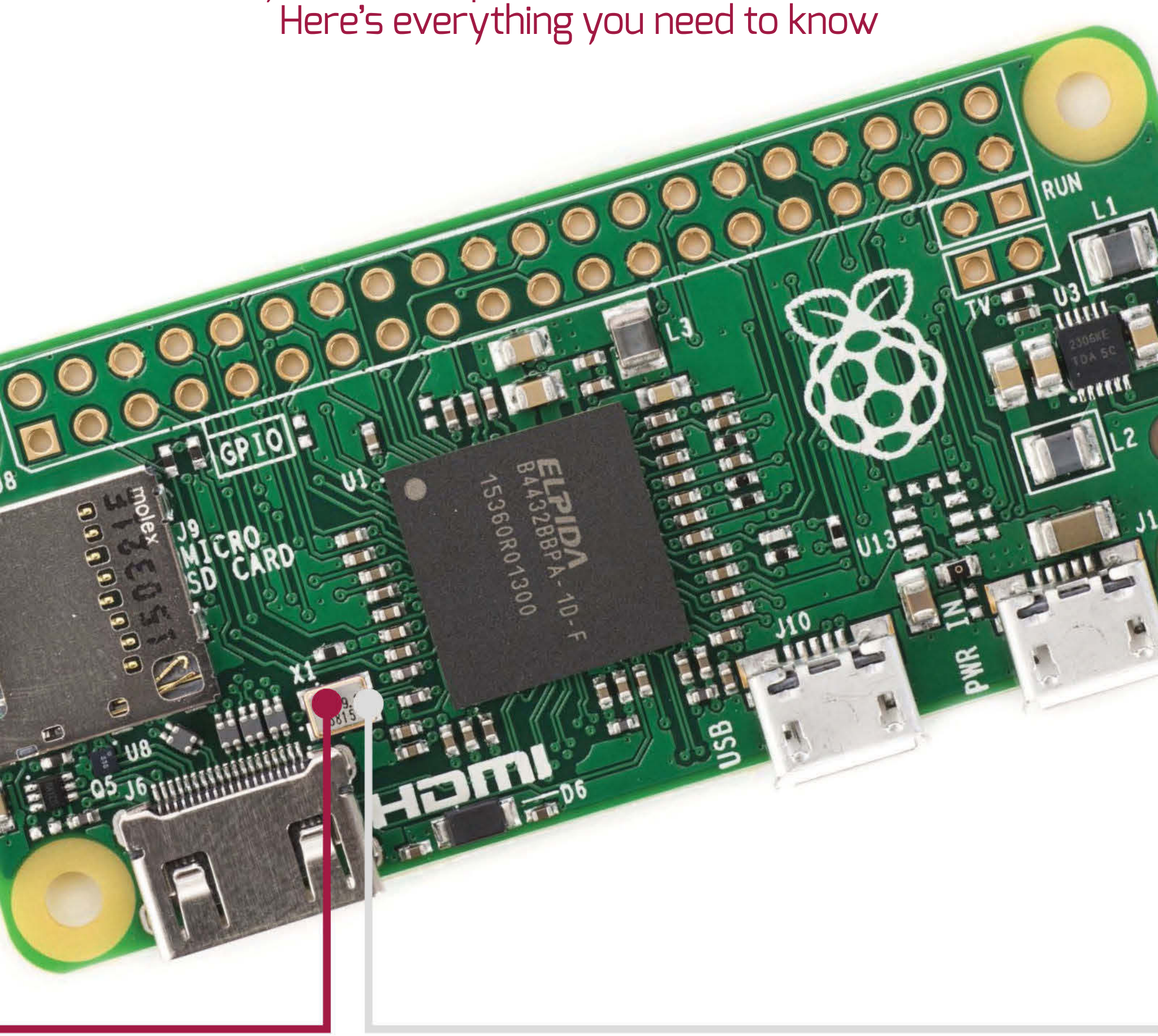
Moniture CPU temperature with Dizmo
Learn to use a multi-touch, widget-based interface

Create a digital assistant – Part 2
J.A.R.V.I.S. can hear. Now let's make it understand

Talking Pi
Your Raspberry Pi questions answered

FAQ Raspberry Pi Zero

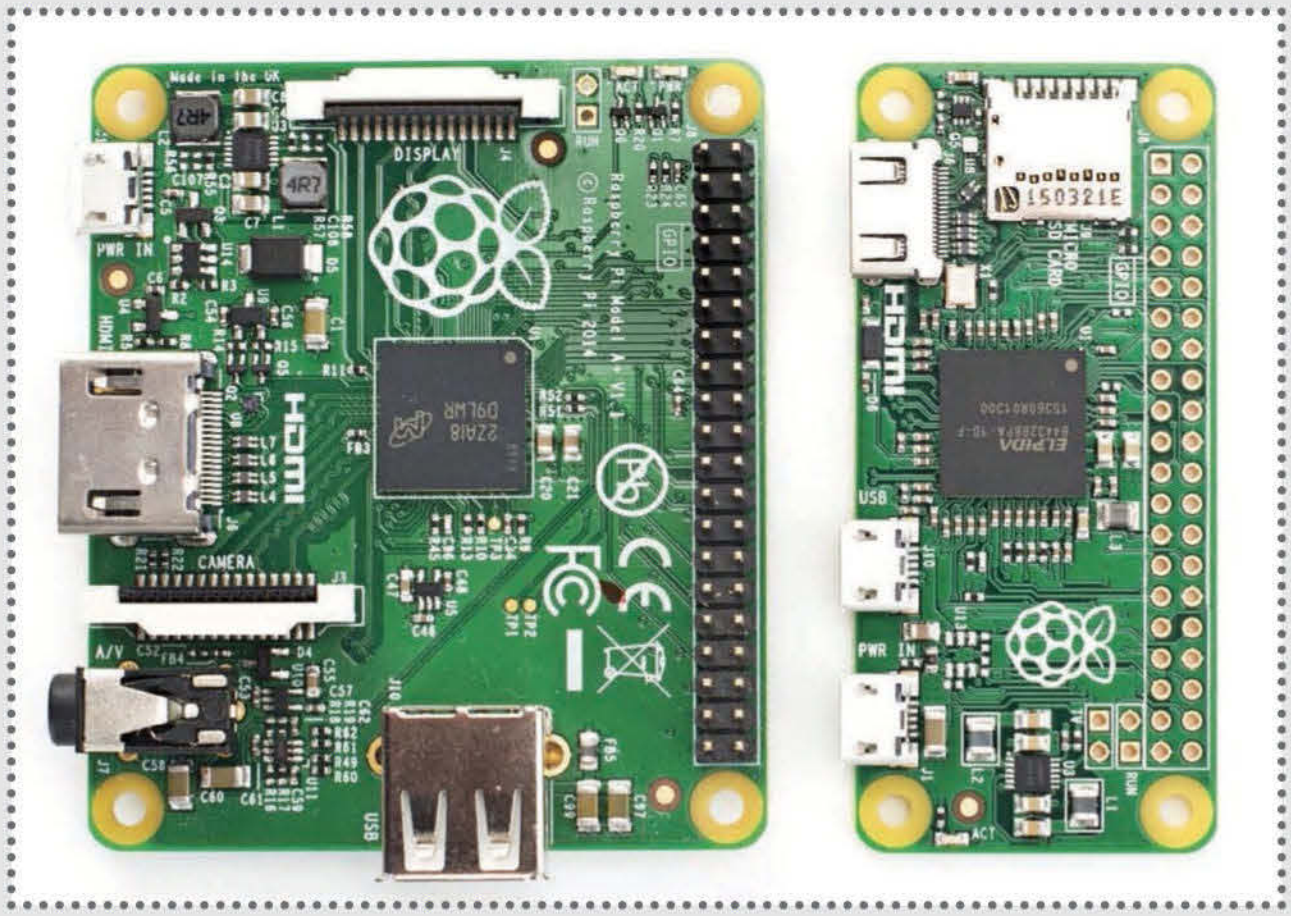
The Raspberry Pi Foundation surprised everyone with an early Christmas present – a brand new board! Here's everything you need to know



What happened to the Raspberry Pi 3?
Well, calling this the Raspberry Pi 3 would imply it's more powerful than the Raspberry Pi 2. It's not really an upgrade – more of an awesome downgrade.

So what does that mean?
The Raspberry Pi Zero isn't as powerful as the Raspberry Pi 2, and a few bits and pieces like the USB 2.0 and Ethernet ports have been removed. The whole point of it is to make the computer as cheap as humanly possible, and by selling it for just $5, the Raspberry Pi Foundation has definitely succeeded!

$5? That is awesome. But it's not as good as other Raspberry Pi models?
Depends what you mean by 'good'. The Raspberry Pi Zero is on about the same level as the original Raspberry Pi Model B – it uses the same processor, rather than the newer BCM2836 that's used in the Pi 2. However, that



Left The Pi Zero is 2.5 inches long and just over an inch wide

"The whole point of it is to make the computer as cheap as humanly possible, and by selling it for just $5, the Raspberry Pi Foundation has definitely succeeded!"

processor has been overclocked and the Pi Zero actually runs about 40 per cent faster than the original Raspberry Pi.

That's cool. Does it have the same RAM as the original Raspberry Pi as well?

It's got 512MB of RAM – the very first Raspberry Pi Model B only came with 256MB, although later versions of the same board were upgraded to 512MB.

So it's a super-cheap, better version of the original Raspberry Pi?

Not quite. As we said, a couple of sacrifices had to be made in order to bring the cost right down. The 40-pin GPIO header on the Pi 2 and B+ is still there, for example, but it's unpopulated.

And there are no USB ports?

Not the full-size ones that you're used to, but there are two micro-USB ports. One is reserved for the power cable, while the other one is free for you to use.

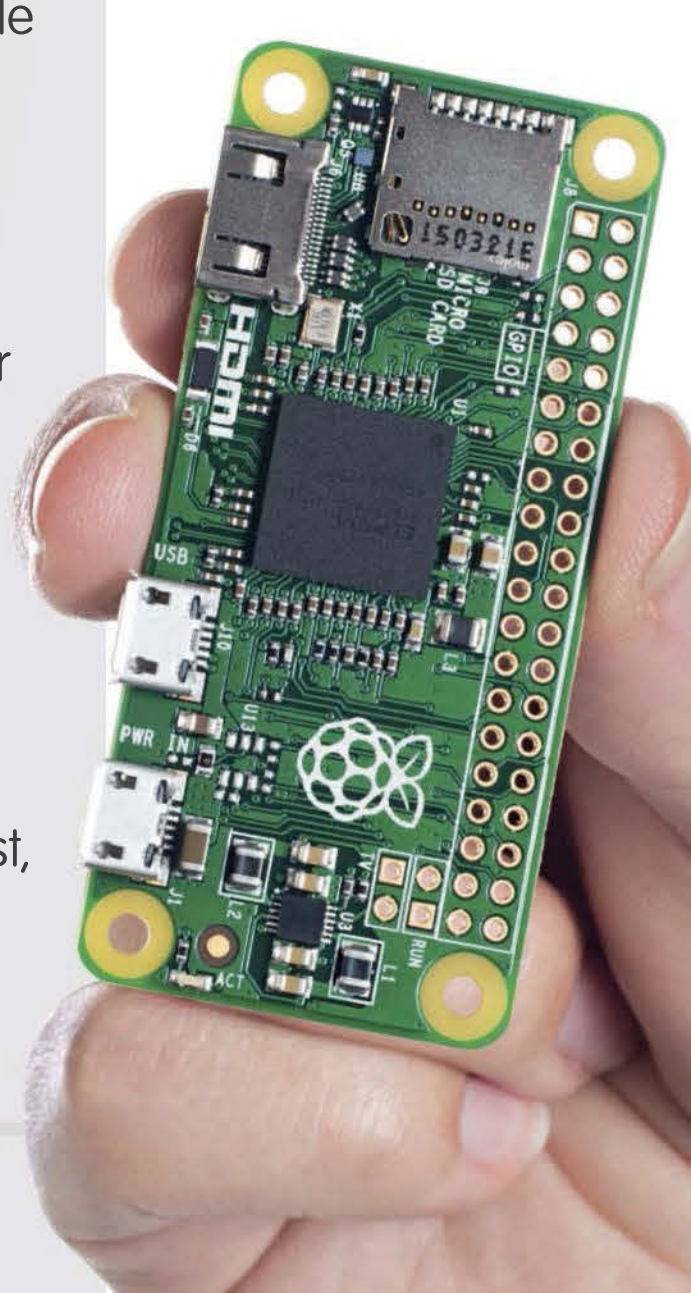
So how can I plug in a keyboard and mouse at the same time?

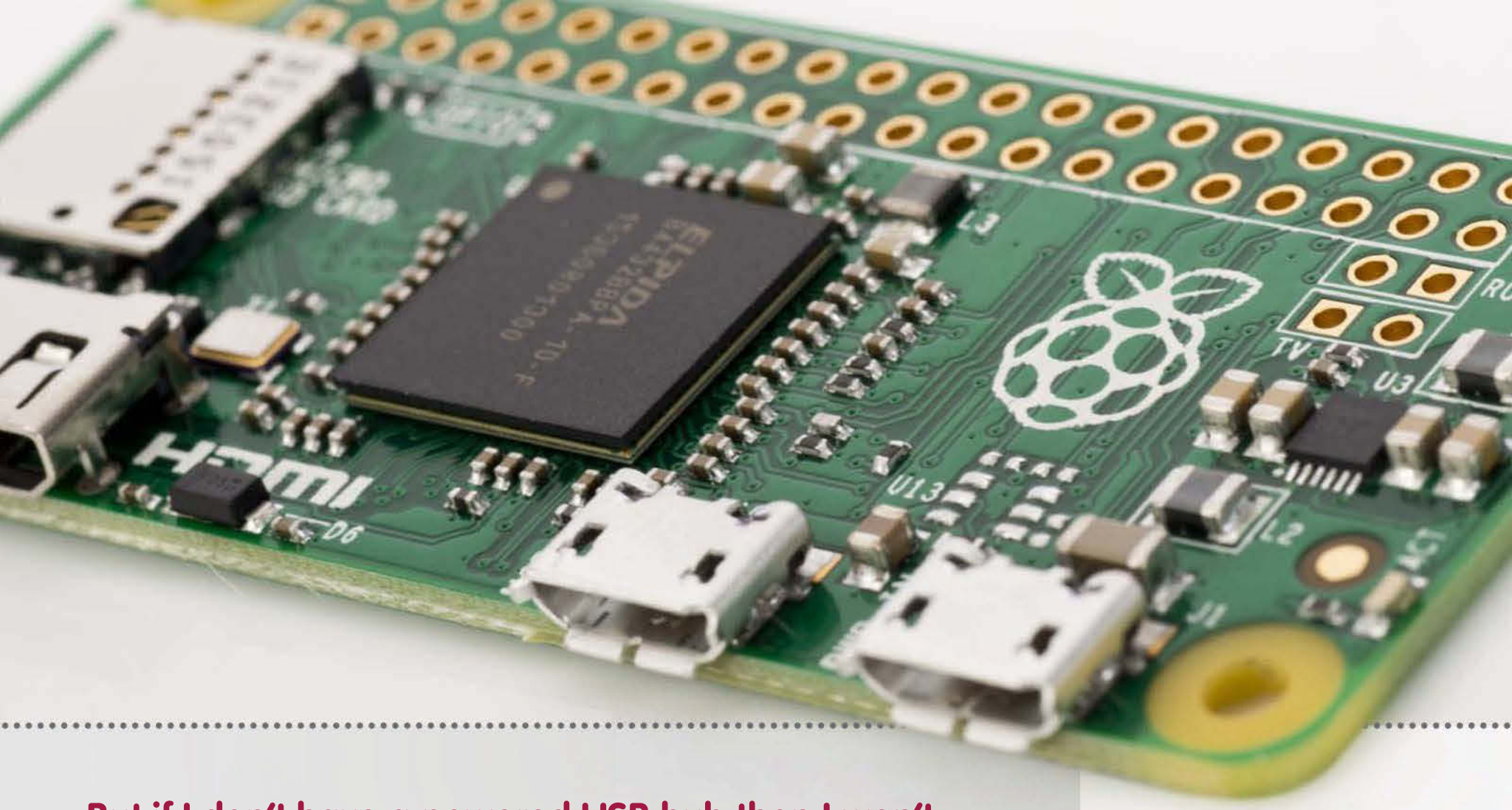
You'll need two things for that: a micro-USB to USB adaptor and then a powered USB hub. Plug the adaptor into the micro-USB port, power your USB hub, then you can plug things like your keyboard and mouse into the hub.

What about the Ethernet cable – is there anything to replace that?

Nope – but if you plug in the micro-USB to USB adaptor first, you can just use your regular Wi-Fi dongle to get online.

"The processor has been overclocked and the Pi Zero actually runs about 40 per cent faster than the original Raspberry Pi"





But if I don't have a powered USB hub then I won't have room for a keyboard and mouse as well, right?
You're absolutely right. Don't forget, though, that if your Pi Zero has a Wi-Fi dongle and is online, you can use SSH to access it from another computer.

And how do I connect my monitor?
You'll need the mini-HDMI port for that, which is in the corner below the microSD slot. You'll also need to grab a mini-HDMI to HDMI cable.

So I need pins for the GPIO header, a cable for the mini-HDMI and an adaptor for the micro-USB. Where do I get all of that?
Where else? The Swag Store! Just point your browser to **http://swag.raspberrypi.org** and you can order the Raspberry Pi Zero Adaptor Kit for just £4.

Ace! So once I've got the hardware, do I need to use a special Raspbian for this?

Above You need to solder a 2x20 pin header onto the unpopulated GPIO connector to use it

Nope – the standard Raspbian distro works just fine with the Pi Zero. We'd recommend a fresh installation, though. The best way is to get a new microSD card and download NOOBS onto it, or just order one with NOOBS already on it from the Swag Store.

NOOBS?

New Out Of Box Software, not the gaming kind. The download links and the instructions are both over at: **www.raspberrypi.org/downloads/noobs**. Basically, you use it to install a Linux distro onto your Pi, like Raspbian.

Sounds great! Anything else I should know about?

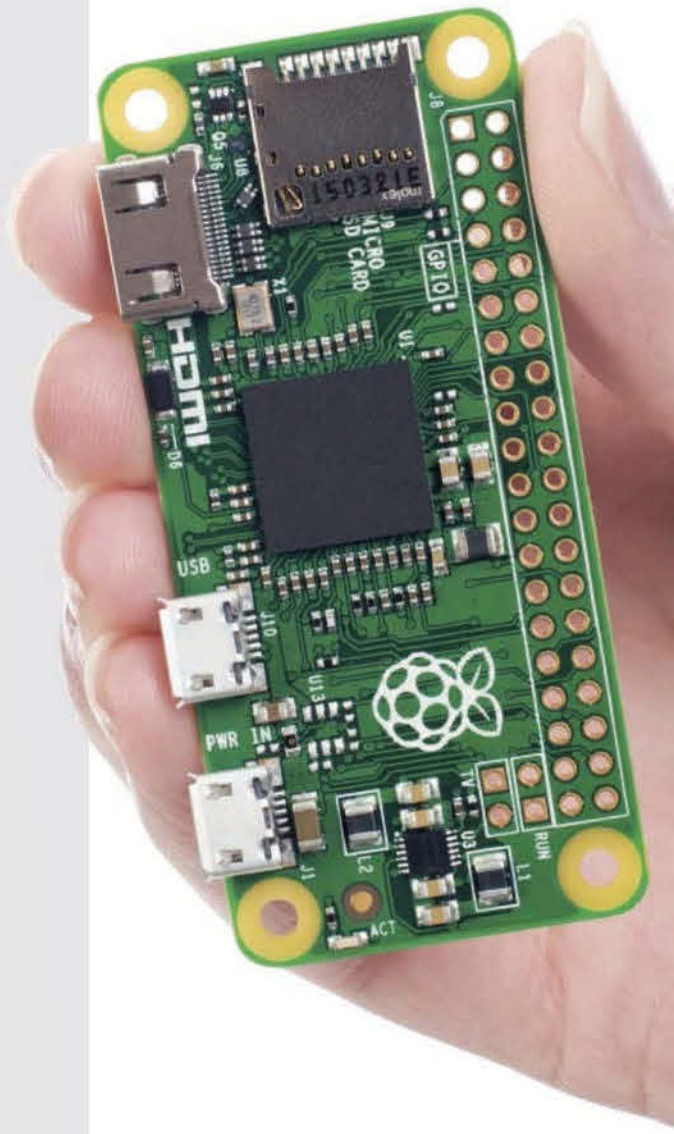
Along with your 40-pin GPIO header, there are also some extra pins that you can solder on: an unpopulated RCA composite video-out pin and an unpopulated RUN mode pin. They're both found just below the far-right end of the GPIO header.

What's the RCA one for?

You use that one to hook the Pi Zero up to older display gear, in case you can't or don't want to use the mini-HDMI. The great thing is that you can even use the RCA composite video-out to connect your Pi Zero to an old TV.

And what about the RUN pins?

Solder these on and you can then connect a reset switch to your Raspberry Pi Zero. This can be really useful if you're testing new projects and find that your Pi is crashing or just hanging for an unreasonable amount of time – if you can't reboot from the command line, you can just use the switch instead. They are also just quite handy generally.



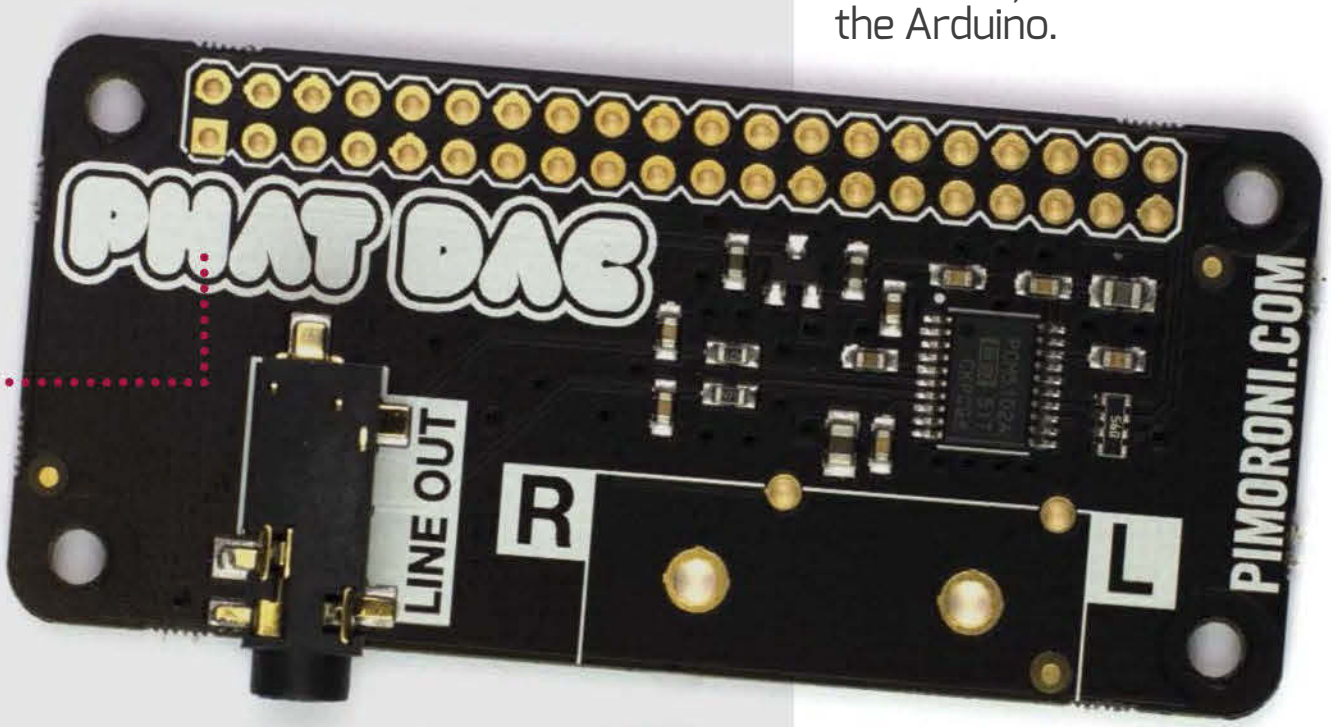
"There are also some extra pins that you can solder on: an unpopulated RCA composite video-out pin and an unpopulated RUN mode pin"

Just one more question – can I use my HATs with this?

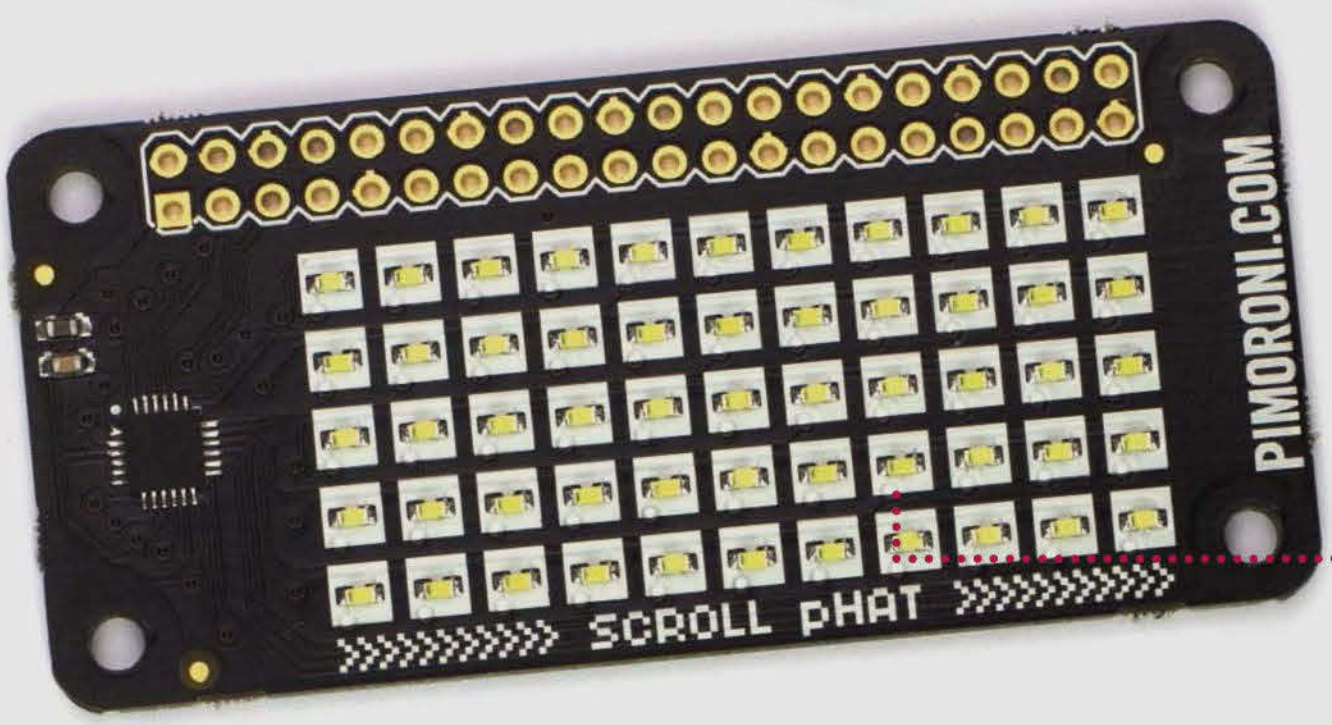
Yep, they'll work just fine. But there are some new ones from Pimoroni (**https://shop.pimoroni.com**) that have been made specifically to fit the Pi Zero – check them out below!



Explorer pHAT, £10
This one provides you with four analogue inputs, two H-Bridge motor drivers, and then four 5V tolerant inputs and four 5V powered outputs. It's perfect for robotics and other projects involving motors, solenoids and relays, as well as working with 5V systems like the Arduino.



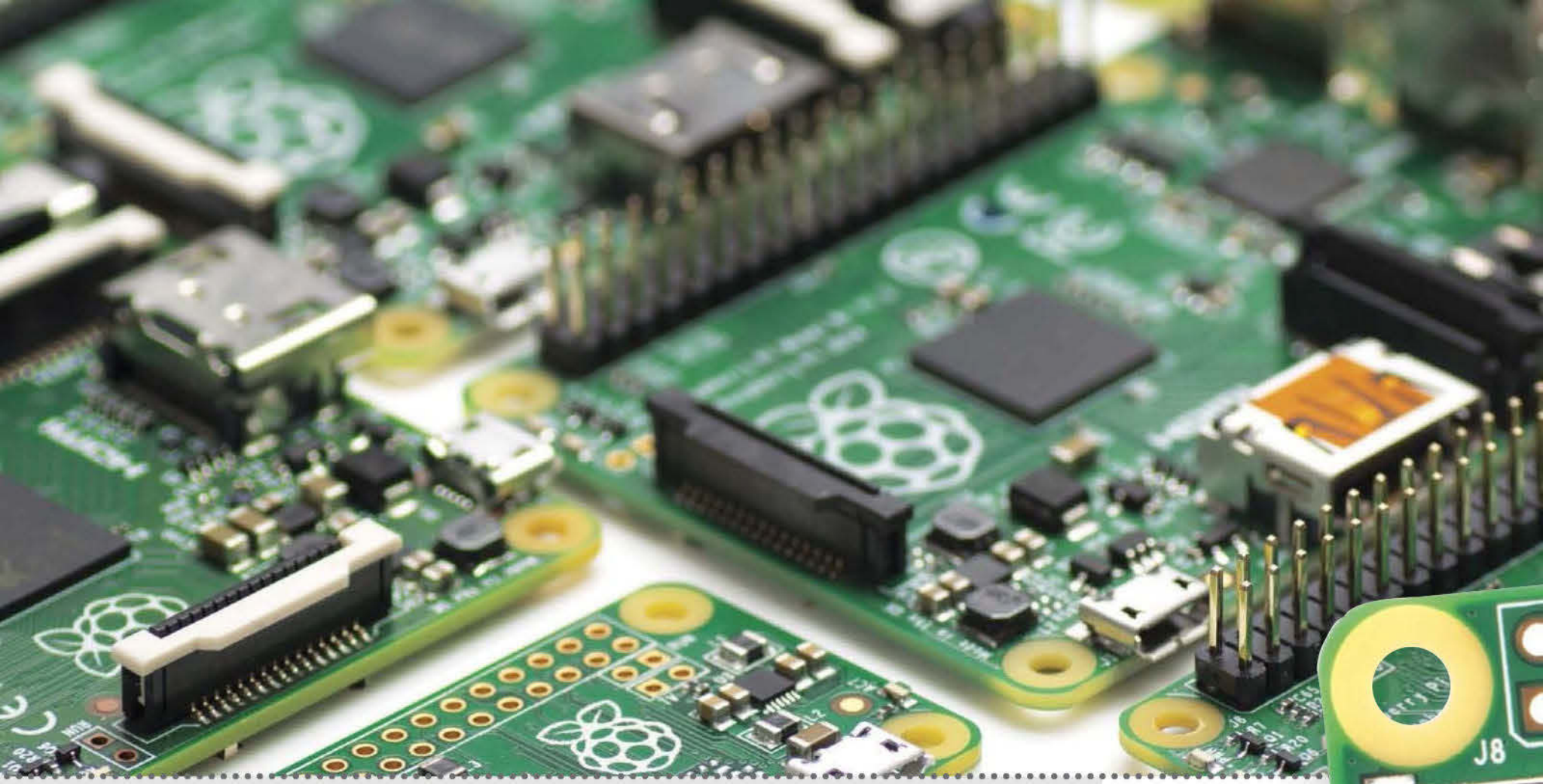
pHAT DAC, £12
Based on Texas Instruments' PCM5102A stereo audio DAC (digital to audio converter) chip, the pHAT DAC adds high-quality audio output to your projects. It pumps out 24-bit high-performance audio at 192KHz – amazingly, your average CD quality is only 16-bit at 44.1KHz!



Scroll pHAT, £10
Perfect for low-power displays, like simple messages and images, the Scroll pHAT has a 55-pixel (11 x 5) matrix of white LEDs with brightness control. It's the type of thing that's useful for displaying system info like CPU usage, or even displaying tweets via the Twitter API.

50 Ways to Master your Pi

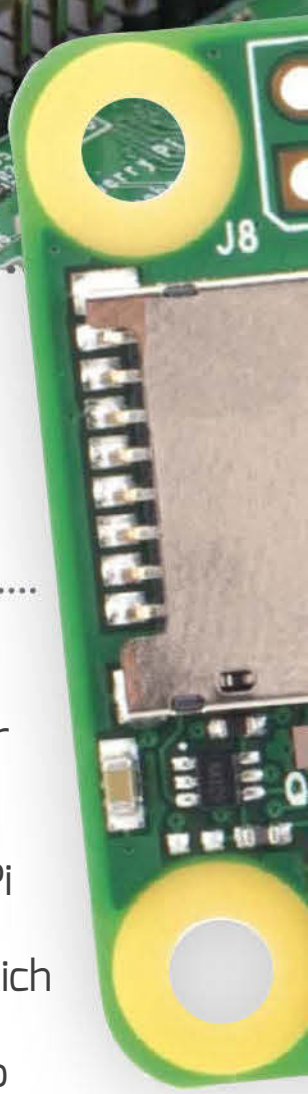
Get the most out of your Pi with these expert tips and tricks



Whether you've just gotten your lucky hands on a powerful, petite new Raspberry Pi Zero or you're looking to maximise the efficiency of the faithful Raspberry Pi you already own, this is everything you need to get started on boosting not only your Raspberry Pi but your own knowledge. The Raspberry Pi is a versatile little piece of hardware, with a wonderfully creative amount of potential, and though most of you will be familiar with its more day-to-day functions, there are always tweaks and adjustments to be explored that can tailor the Raspberry Pi to your own desired user experience.

From soldering to useful Python features, GPIO interrupts to remote access, and a whole lot more, this masterclass in technical and practical skills covers fifty useful ways to get the most out of your Raspberry Pi. Whatever your skill level, there's something here for you to get your teeth into. Plus, we've supplied a few useful Python scripts that tie in with some of these tips – grab them now from: **http://bit.ly/1ZbH5Qy**.

Pi Zero



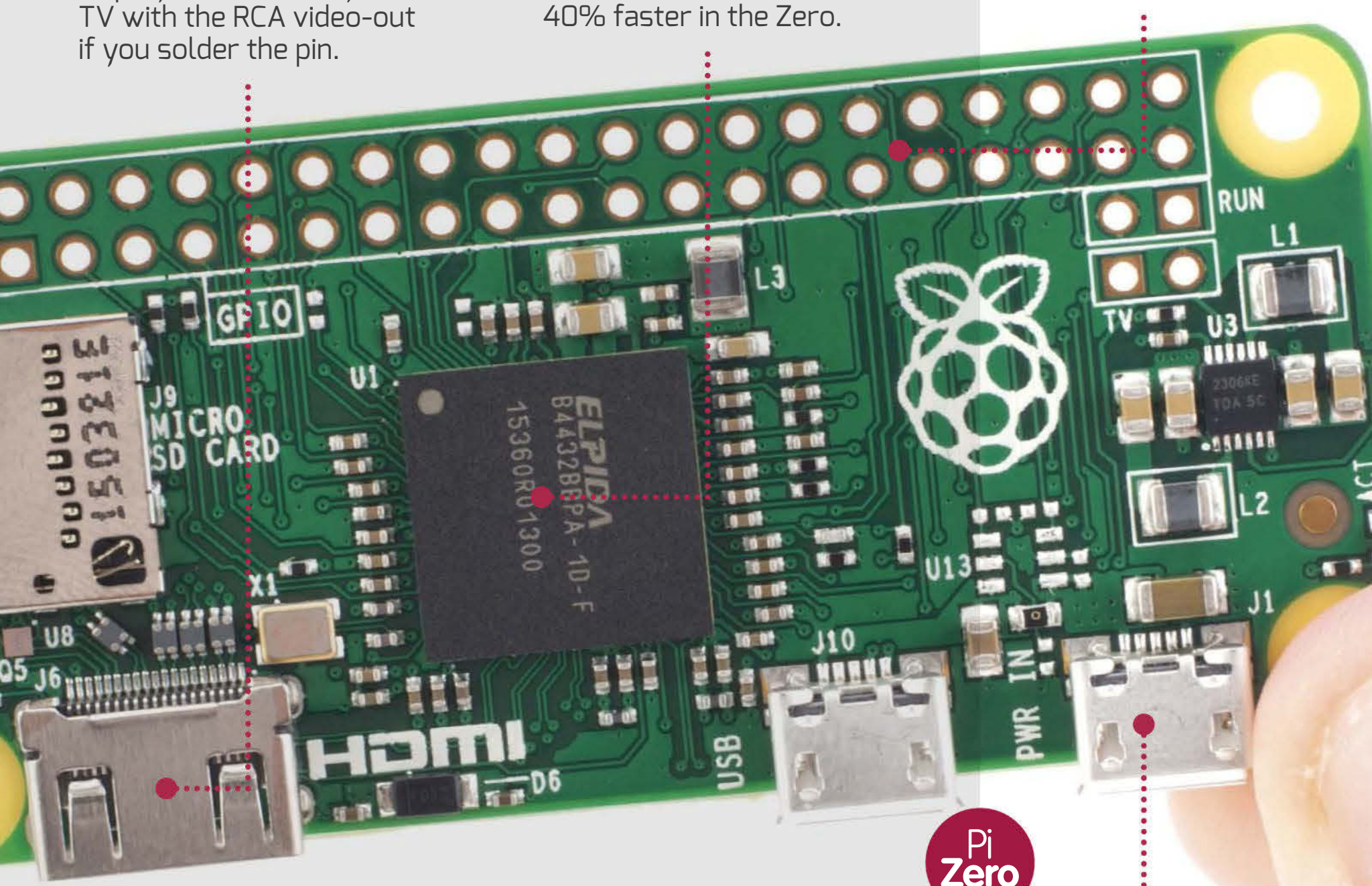
Every single tip here will work on your Pi Zero; just keep an eye out for the 'Pi Zero' flash, indicating which are relevant to the Pi Zero only. Those of you using an earlier model won't be missing out, though – how could we ever neglect our favourite single-board computer? – the majority of tips, tricks and tweaks are still suitable for any other official Raspberry Pi as well. Have fun tinkering!

01 Raspberry Pi Zero

Mini-HDMI You'll need a mini-HDMI to HDMI cable to use your monitor as a display. You can use your TV with the RCA video-out if you solder the pin.

BCM2835 This processor was used in the original Raspberry Pi models, but it's been overclocked to run at 900MHz and is about 40% faster in the Zero.

Pinless port The GPIO header is the same as in the newer models but comes without the pins. You'll need to solder on a 40-pin header block.



The newest member of the Raspberry Pi family, this tiny board is the result of the Raspberry Pi Foundation's efforts to reduce the cost of the computer even further. Not content with reducing its $35 computer to $25, which can still be a little pricy for some people, it has cut it right down to $5 by making a few adjustments. Here's what you need to know.

Micro-USB One of these ports is for your micro-USB power supply. To use peripherals and a Wi-Fi dongle, you'll need a micro-USB to USB adaptor so you can attach a powered USB hub.

Vital knowledge

02 Find your Pi on a network

If you can't log into your router to view DHCP leases, you can use nmap (available for Linux, Windows and Mac) to scan the local network to find a Raspberry Pi. You need to know the address range of your local network (common networks are 192.168.1.0/24, and 192.168.2.0/24). **nmap -sn 172.17.173.0/24** will run a ping scan and output a list of devices. An example output is:

```
Nmap scan report for raspberrypi.home.org
    (172.17.173.21)
Host is up (0.0011s latency).
```

03 Experiencing stability issues?

By far the biggest cause of stability issues is the power supply you are using, especially now that the Raspberry Pi 2 has more CPU cores and so uses more energy. The recommended current supply for a Raspberry Pi 2 or B+ is 1.8 amps. If you are still having issues, your SD card may be worn out.

04 Enable max current draw

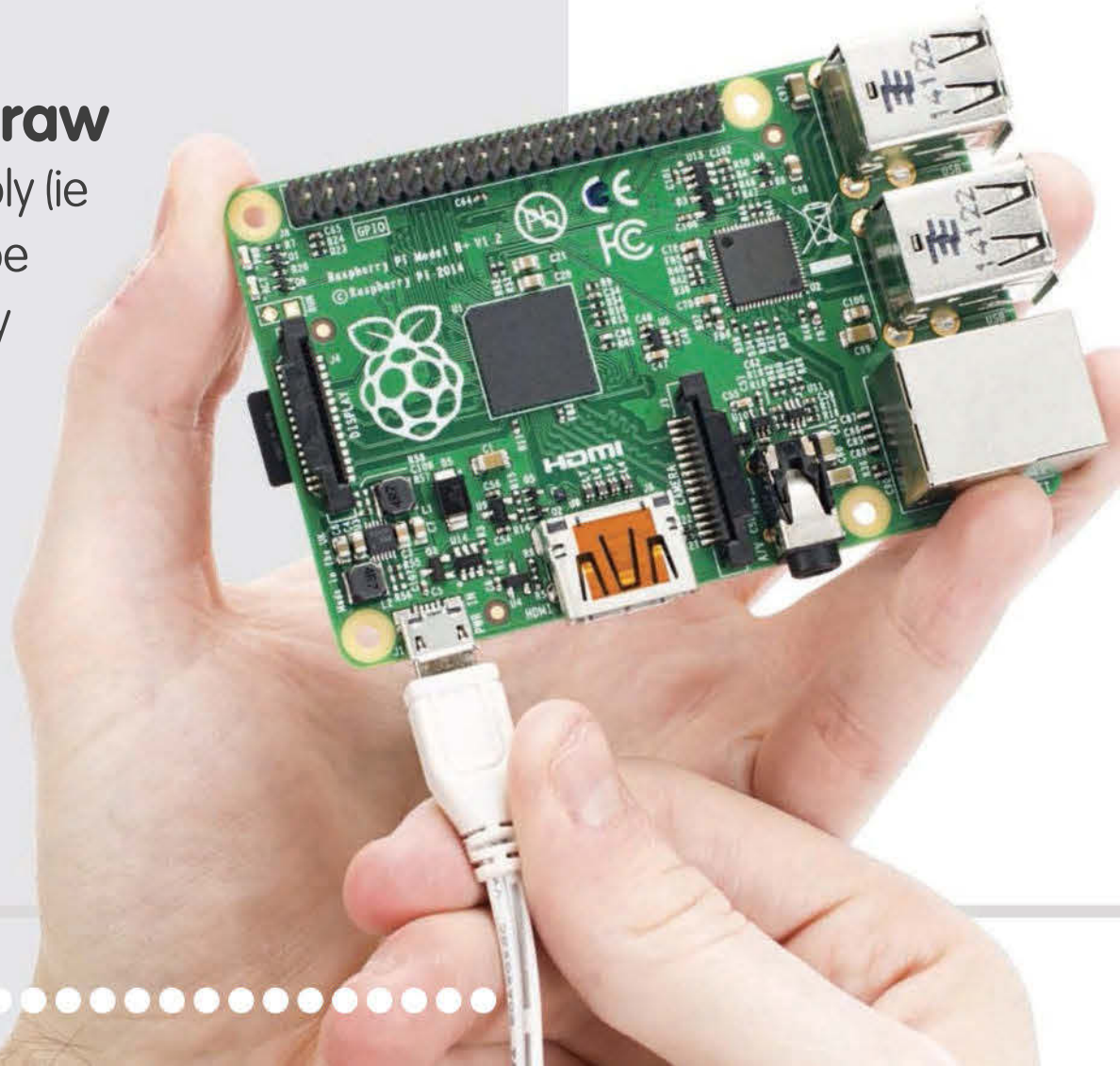
If you have a good power supply (ie 2 amps or more) and want to be able to connect a current-heavy device to your Pi 2 or B+, such as a USB hard disk, then you can add the line **max_usb_current=1** to /boot/config.txt, which will set the max current over USB to 1.2 amps instead of the default 600mA.

05 Updating

Ensure you have the latest packages on Raspbian by running **sudo apt-get update && sudo apt-get upgrade** in a terminal

06 Forgot to type sudo?

Sudo is used to get root privileges for a specific command (like editing a file in /boot or /etc). The variable **!!** in bash is used to represent the previous command that you typed. So if you need root privileges for something but forgot to type sudo then you can simply type **sudo !!**, provided you haven't typed in anything else since then.



07 Get Pi Zero online

Pi Zero

Step One The parts

You'll need an adapter to connect the micro-USB port to a full size USB port. You can get the adapter, plus a mini-HDMI to HDMI adapter and GPIO pin headers, at: **http://swag.raspberrypi.org/products/pi-zero-cables**. You'll also need a USB to Ethernet adapter that works with Linux (most will work out of the box), These can easily be found on Amazon for around £10.

Step Two Configuration

Since the Pi Zero uses the normal Raspbian image, and eth0 (ie the built-in Ethernet card) is missing, there is no configuration necessary because the USB Ethernet card will take the missing LAN chip's place as eth0.

Step Three Testing it out

If the activity lights on your USB to Ethernet adapter are lit up, then it should be working fine. You can now use the remote access tips that are just two swipes forward from this page with your Raspberry Pi Zero!

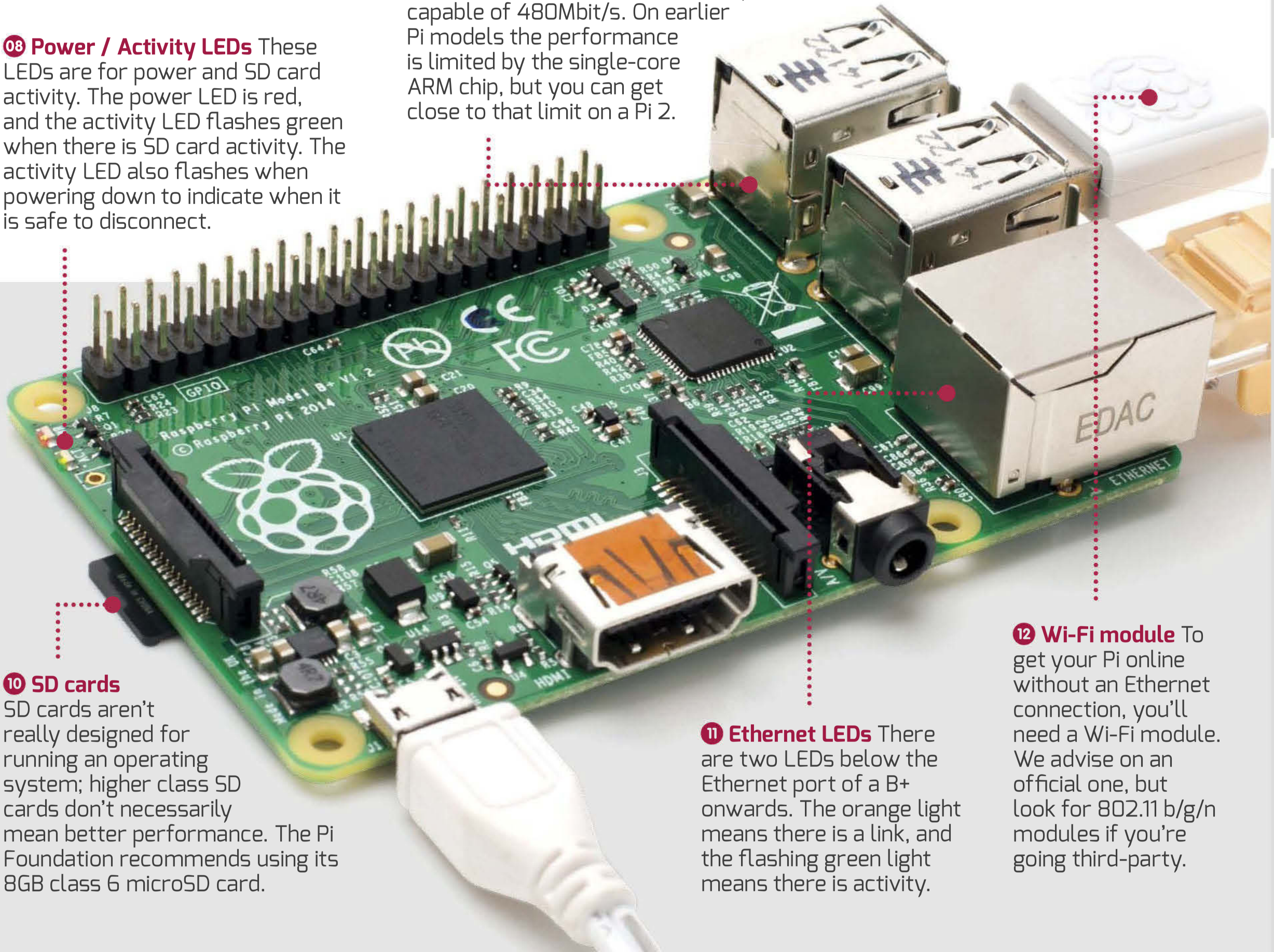


Left The Swag Store's Adaptor Kit contains a male GPIO header, but you can solder on a female header instead if you want to

Models B+ and 2B

08 Power / Activity LEDs These LEDs are for power and SD card activity. The power LED is red, and the activity LED flashes green when there is SD card activity. The activity LED also flashes when powering down to indicate when it is safe to disconnect.

09 USB controller The USB controller on the Pi is theoretically capable of 480Mbit/s. On earlier Pi models the performance is limited by the single-core ARM chip, but you can get close to that limit on a Pi 2.



10 SD cards SD cards aren't really designed for running an operating system; higher class SD cards don't necessarily mean better performance. The Pi Foundation recommends using its 8GB class 6 microSD card.

11 Ethernet LEDs There are two LEDs below the Ethernet port of a B+ onwards. The orange light means there is a link, and the flashing green light means there is activity.

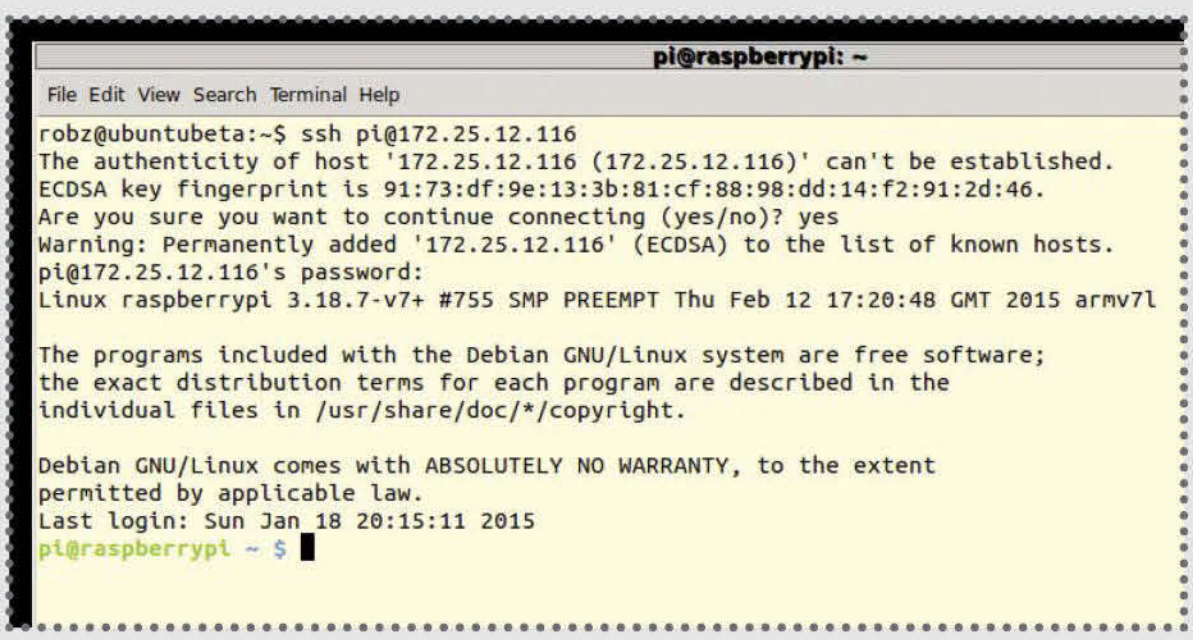
12 Wi-Fi module To get your Pi online without an Ethernet connection, you'll need a Wi-Fi module. We advise on an official one, but look for 802.11 b/g/n modules if you're going third-party.

Transferable tips

Handy hints get the most out of any Pi model

13 Remote access with SSH

SSH stands for Secure Shell. You can use SSH to connect to a terminal session on your Raspberry Pi over your local network. This means that you can use the Pi with only a network cable and power cable connected, which is much more convenient than needing a screen and keyboard. Once you have found your Pi on the network you can log into it using the default username of 'pi' and the default password of 'raspberry'. Both Linux and Mac will have built-in SSH clients, so simply open the terminal and type **ssh pi@192.168.1.5**, assuming that 192.168.1.5 is the address of your Pi. On Windows, you can use an SSH client called PuTTY, which is free to download, doesn't need installing and is easy to find with a search engine.



14 Copy files using SCP

SCP stands for Secure Copy Protocol, and is a way for you to copy files (and directories) to and from your Raspberry Pi over the network. A good use of this would be if you have art for a Pygame project and you need to copy it over. FileZilla is a decent graphical SCP client to

> "You can use SSH to connect to a terminal session on your Pi over your local network. This means that you can use the Pi with only a network cable and power cable"

15 Control options

sudo raspi-config can be used to change several options. For example, to enable the camera, overclock the Pi, change boot options, and expand the filesystem to use the full space on the SD card.

use (connect to your Pi on port 22 with the username 'pi' and password 'raspberry'). If you are using SCP from the terminal then the syntax is as follows:

```
scp -r testdir/pi@172.17.173.21:~/
```

The **-r** switch is for recursive, which means entire directories are copied in addition to files. This will place your test directory in /home/pi/ (because **~** points at the logged-in user's home directory on Linux). Simply swap the syntax for copying from the Raspberry Pi instead:

```
scp -r pi@172.17.173.21:~/testdir .
```

The dot refers to the current directory in Linux, so the testdir directory would be copied to the current directory.

16 Set up a VNC server

VNC stands for Virtual Network Computing. Using VNC, you can access the Raspbian desktop over the network (meaning you only need power and Ethernet connected). There is no audio support but for any other tasks, VNC should provide acceptable performance. You can install a VNC server with the following commands:

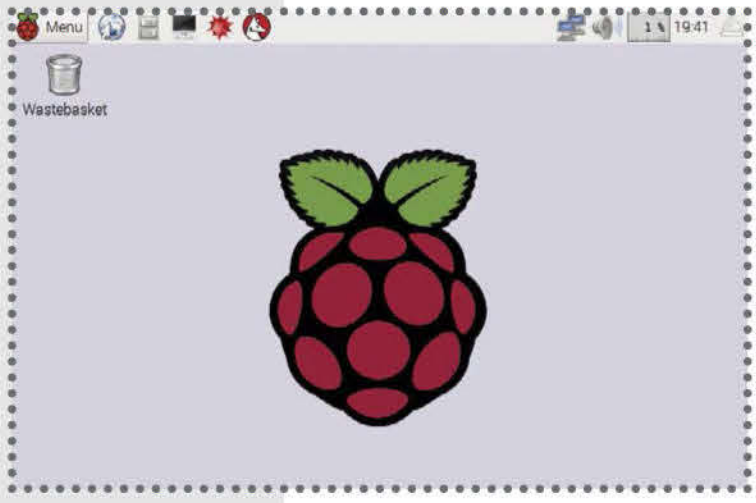
```
sudo apt-get update
sudo apt-get install tightvncserver
```

There are several free VNC clients available, such as VNCViewer. To start a VNC session on your Pi, log in over SSH and then run **tightvncserver**. You will be prompted to enter a password the first time you run it. You can specify a screen resolution with the -geometry option, for example **-geometry 1024x768**. You can kill an existing VNC session with **tightvncserver -kill :1**, where 1 is the session number. To connect to that session on a Linux machine, you could use the command: vncviewer 172.17.173.21:1, substituting for the IP address of your Pi.

17 Cache flush

The **sync** command ensures that everything has been flushed to permanent storage from the cache. It can be useful to run after updating packages, for example, or changing your network configuration.

Below Access the Raspbian desktop from your main computer over your local network



Five terminal tricks

18 Using wget to download files

Wget can be used to download files from the Internet from the terminal. This is convenient if you need to download a zip file containing source code and then extract it. For example:

```
wget http://liamfraser.co.uk/test.zip
unzip test.zip
```

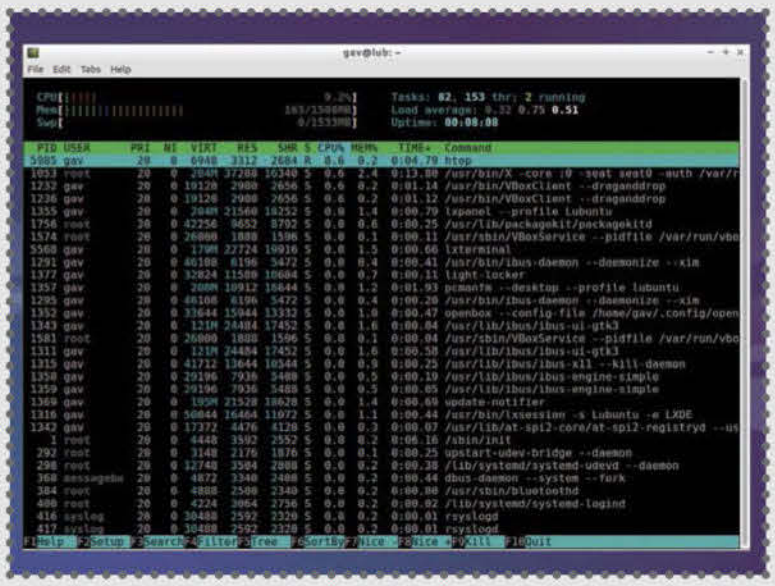
19 Using screen

Screen (available via apt-get) is great if you have something you want to run that takes a long time. You can run it in a screen session, detach from it and reconnect later on. Example usage:

screen -S test to start (Ctrl+A, d to disconnect)
screen -ls to list sessions.
screen -r test to reconnect
exit will kill bash and therefore end the screen session.

20 Using htop to monitor load

Htop is an improvement on the original top utility. It lets you see the CPU and memory load on your machine in real time, which is useful to know how intensive your application is being on a Raspberry Pi. You can install it with **sudo apt-get install htop**.



21 Listing open ports

Lsof stands for List Open Files, and you can install it with **sudo apt-get install lsof**. If you run **sudo lsof -i** you will see a list of all open network connections on the machine, including applications that are listening for new connections (for example, the SSH daemon).

22 Reboot from the terminal

It seems like a simple tip but not everyone knows that you can reboot your Pi with the command **sudo reboot**, and similarly power it off with **sudo poweroff**. You need to disconnect and reconnect the power after powering off the Pi, though, as there is no on/off switch.

Left Htop has a cursor-controlled interface and is a little easier to work with than the top utility

Become a Python Pro

Make your code stand out with these pointers

23 The main function

Having a main function (if __name__ == __main__) in Python is useful. It makes it easier to see the difference between where your functions/classes are defined and where your entry point is. More importantly, it allows you to only run code from that Python script in the case that it is the main script being run. This means that you can create a class and have code that tests that class to make sure it works. However, if you were to include your file in another program, the code in the main method would not be run and you can just use the class that you require.

```
class MyClass:
  def __init__(self):
    self.x = 1
  def print_increment(self):
    print “x = {0}”.format(self.x)
    self.x += 1
if __name__ == “__main__”:
  # Test MyClass
  m = MyClass()
  m.print_increment()
  m.print_increment()
```

24 Command line arguments

Command line arguments enable your program to run in various modes depending on the options that the user passes to the program when running it. Command line arguments in Python are given in sys.argv, which is a list of arguments. The first argument is always the

“Command line arguments in Python are given in sys. argv, which is a list of arguments”

25 Split files

If your program is large, you can split it up into multiple files. If you have a file called MyClass.py containing a class MyClass, you can use it from another script with the line **from MyClass import MyClass**.

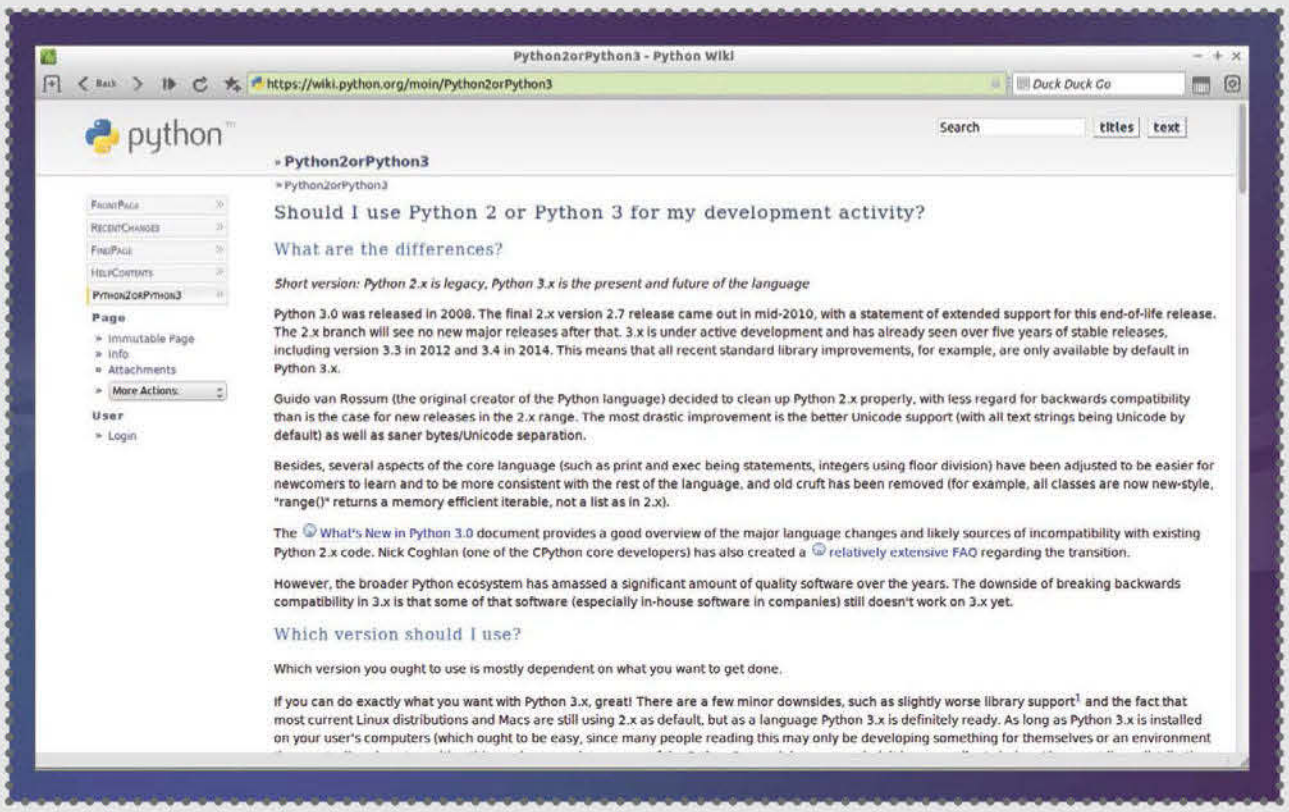
name of the script that has been executed. For example, some simple code that prints the argv array produces the following outputs:

```
$ python args.py
['args.py']
$ python args.py --foo
['args.py', '--foo']
```

You can check for command line arguments in this list. If the length of the list is 1 and you require arguments, print a help message with instructions.

```
import sys

help_msg = "My help message"
debug_mode = False
if __name__ == "__main__":
    if len(sys.argv) == 1:
        print help_msg
        sys.exit()
```



"You can check for command line arguments in this list. If the length of the list is 1 and you require arguments, print a help message with instructions"

Left Not sure whether to pick Python 2 or 3? There's a guide to help in the docs: **http://bit.ly/1jyd799**

```
  if “--debug” in sys.argv:
     debug_mode = True
     print “Using debug mode”
```

26 Using properties and setters

Properties in Python are a way to write getters and setters for variables. You need new-style classes (you need to inherit from object). We have created a class called distance where the distance in miles is a variable, and the distance in kilometres is a property. Getting from that variable multiplies the distance in miles to be kilometres and setting the variable sets the value in miles.

```
class Distance(object):
  KM_PER_MILE = 1.60934

  def __init__(self, mi):
     self.miles = mi
  @property
  def km(self):
     return self.miles * Distance.KM_PER_MILE

  @km.setter
  def km(self, value):
     self.miles = value / Distance.KM_PER_MILE

if __name__ == “__main__”:
  d = Distance(3.1)
  print d.km
  d.km = 10
  print d.miles
  print d.km
```

27 Recent items

You can use negative indexes on Python lists to get the most recently added item(s). **mylist[-1]** will get the latest thing that was added to the list.

28 Using GPIO in Python

Step One Import library

Start by importing Rpi.GPIO:

```
import RPi.GPIO as GPIO
```

Then, you want to set the pin numbering convention to the Broadcom mode – as in GPIO17 will be pin 17, rather than being pin 11 on the Pi:

```
GPIO.setmode(GPIO.BCM)
```

Step Two Pin setup

Now you need to set up your pins as either inputs or outputs with the following syntax:

```
GPIO.setup(5, GPIO.OUT)
GPIO.setup(6, GPIO.IN)
```

If you have several pins to set up, it makes sense to put them in a list and then do something like:

```
for p in pins:
    GPIO.setup(p, GPIO.OUT)
```

Step Three Get values

Once the pins are set up, getting values from them is easy. To get the value of a pin (0 for low, and 1 for high), use the following syntax:

```
value = GPIO.input(6)
```

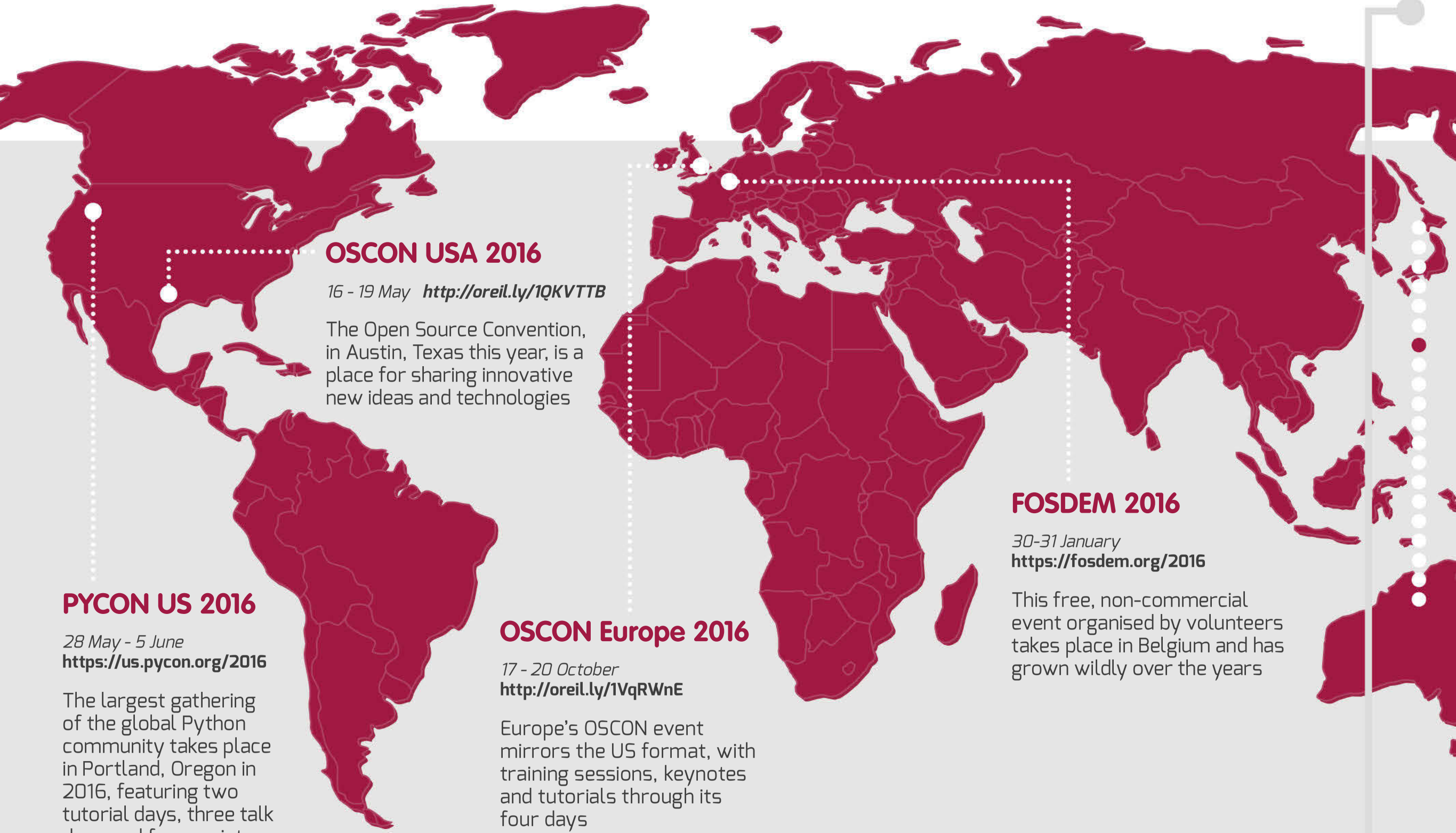
…and to set the value of a pin (with either 0 or 1, or True or False) use the following syntax:

```
GPIO.output(5, True)
```

Please note that if you are starting with Raspbian Jessie, you shouldn't need sudo to access the GPIO pins, but in previous versions of Raspbian you will need to use sudo to run your code.

"You want to set the pin numbering convention to the Broadcom mode – as in GPIO17 will be pin 17, rather than being pin 11 on the Pi"

29 Python conferences



OSCON USA 2016

16 - 19 May ***http://oreil.ly/1QKVTTB***

The Open Source Convention, in Austin, Texas this year, is a place for sharing innovative new ideas and technologies

PYCON US 2016

28 May - 5 June
https://us.pycon.org/2016

The largest gathering of the global Python community takes place in Portland, Oregon in 2016, featuring two tutorial days, three talk days and four sprint days for attendees

OSCON Europe 2016

17 - 20 October
http://oreil.ly/1VqRWnE

Europe's OSCON event mirrors the US format, with training sessions, keynotes and tutorials through its four days

FOSDEM 2016

30-31 January
https://fosdem.org/2016

This free, non-commercial event organised by volunteers takes place in Belgium and has grown wildly over the years

Five hidden features

30 Running your script from a terminal

Start your Python script with **#!/usr/bin/env python** and then mark it as executable. You can then execute it from bash like a normal script without typing **python** before it:

```
$ echo '#!/usr/bin/env python' > test.py
$ echo 'print "Hello World from Python!"' >> test.py
$ chmod +x test.py
$ ./test.py
Hello World from Python!
```

31 Assertions

Assertions are useful when writing algorithms. They are used to check the data is valid before and after your algorithm. So, for example, if you are expecting your results in a certain range you can check them. The syntax is **assert(boolean expression)**. For example:

```
>>> assert(0 < 1)
>>> assert(1 < 0)
Traceback (most recent call last):
    File "<stdin>", line 1, in <module>
AssertionError
```

32 Throwing exceptions

It's useful to throw exceptions in your code if it's possible that it can go wrong. That way the person calling your code can put it in a try-catch block and handle the exception if it is raised. If the caller does not handle it then their application will crash.

```
>>> raise ValueError("Supplied value out of range")
Traceback (most recent call last):
    File "<stdin>", line 1, in <module>
ValueError: Supplied value out of range
```

34 Collect data into lists

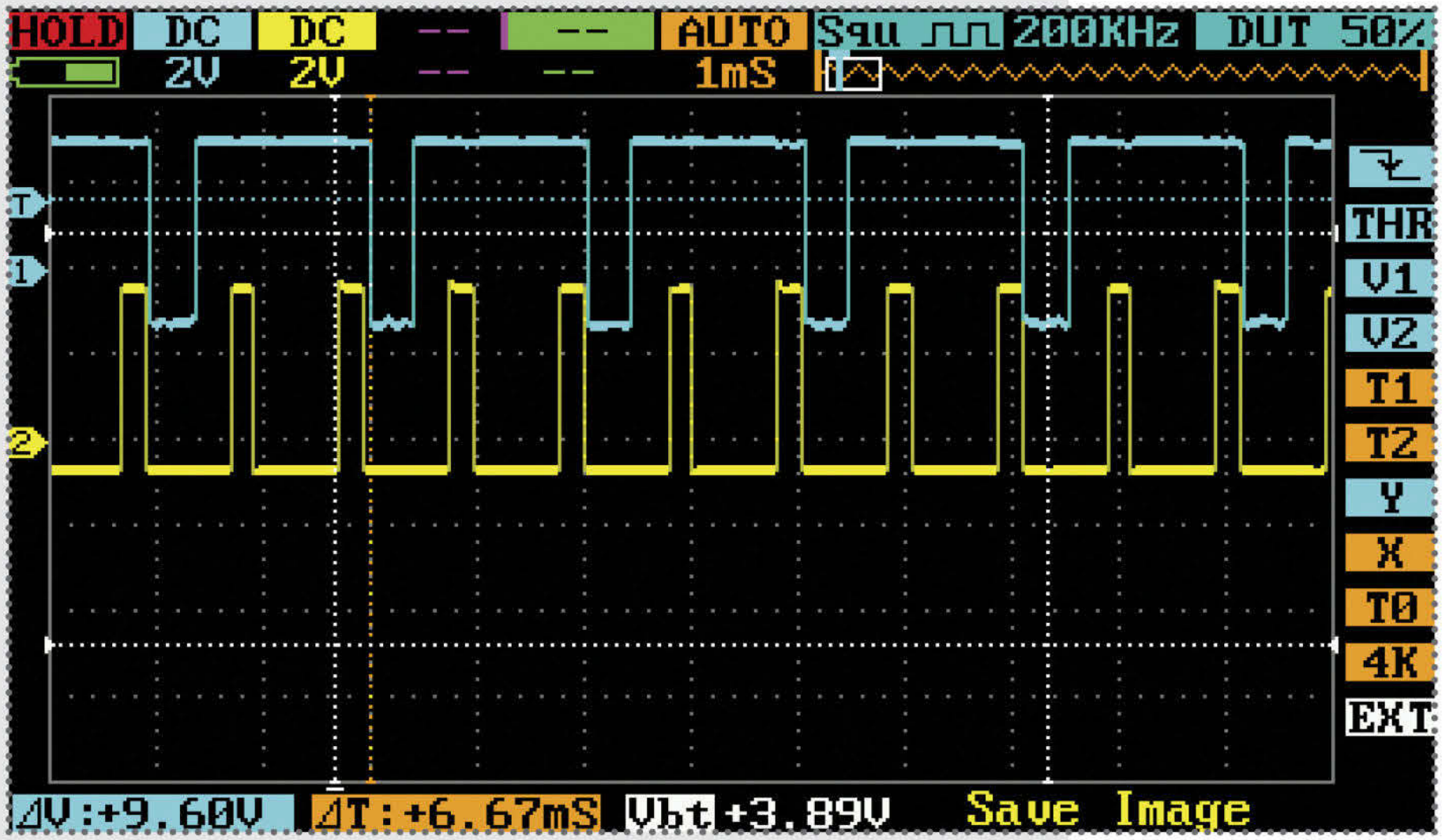
List comprehension is a way of generating a list on a single line. You can use list comprehension to extract values from other list-type data structures. For example, if you have a collection of distance classes, you could get just the distance in miles into a list: **miles = [x.miles for x in distances]**.

34 Using the interpreter

Did you know you can start a Python interpreter to test things are working without having to write a Python script? Just type **python** into the terminal and then simply start writing Python code.

Hardware how-to

GPIO interrupts, PWM, soldering and more



35 Pulse width modulation

Pulse width modulation is where the output of a GPIO pin is high for a percentage of time and low for the remaining percentage of time. The percentage where the pin is high is called a duty cycle. Pulse width modulation is very useful in electronics, especially when it comes to tasks like controlling the brightness of LEDs. To do this in Python:

```
GPIO.setup(5, GPIO.OUT)
p = GPIO.PWM(5, 50) # Frequency of 50 hz
p.start(50) # 50 percent duty cycle
# Do work or wait here so program doesn't exit
p.ChangeDutyCycle(70) # 70 percent duty cycle
p.stop()
```

Above Here are two signal waves with duty cycles of roughly 80% (top) and 20% (bottom)

36 Remove static

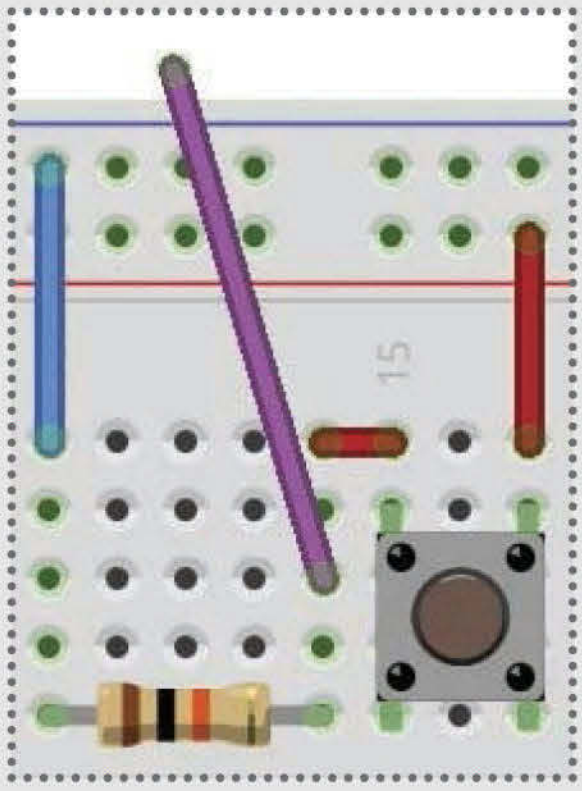
Make sure you haven't built up any static charge when working with electronics. Touching a grounded radiator can help get rid of static charge, but an anti-static wrist strap is better.

37 GPIO interrupts

An interrupt is when a hardware event triggers an interrupt on the CPU, causing it to stop what it is dealing with and run an interrupt request handler. The Raspberry Pi can trigger inputs when a GPIO pin goes high (ie from 0V to 3.3V) or low (from 3.3V to 0V). This can be more efficient than polling for the state of a GPIO pin, as you only have to deal with the pin changing when it happens. Plus, it can actually simplify the flow of your code. The use of interrupts requires root privileges so you will have to run your code with sudo. The code provided (see the first page of this feature) demonstrates how to set up a callback function to deal with a rising edge.

38 Using a push button

Refer to the circuit diagram below. When the push button is pressed, the left pins are connected to the right pins. By using a 10K pull-down resistor connected to ground, the purple output wire (connected to a GPIO pin configured as an input) defaults to 0V when the button is not pressed. The right-hand side of the button is connected to 3.3V, so when the button is pressed, the left-hand side of the button will also be connected to 3.3V. The left-hand side is connected in parallel with the purple signal wire, and also the 10K resistor to ground. Because electricity takes the path of least resistance, the purple signal wire will output 3.3V.



Left The blue wire is the ground, the red one is 3.3V and purple is for the output

39 Safe voltage

Ensure you are only using 3.3V voltage levels when working with the GPIO pins. Connecting anything higher than 3.3V to a GPIO pin will likely damage your Pi.

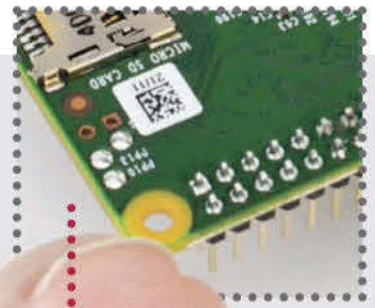
GPIO explained

40 GPIO orientation The best way to verify you have the Pi oriented the right way is to flip it over. The underneath of pin 1 has a square hole for the solder instead of a circular hole.

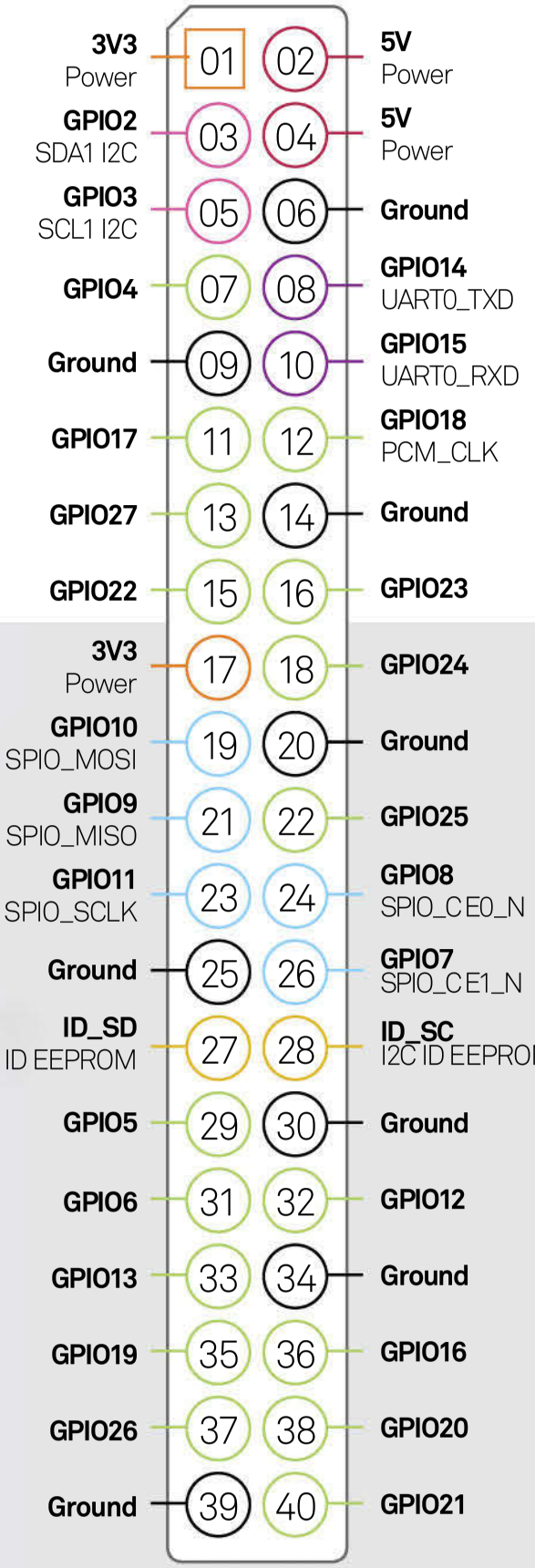
41 Serial console If you connect the UART0_TXD pin (#8) to the receiver pin of a USB-to-serial converter, and the UART0_RXD pin (#10) to the transmitter pin of that converter, you can set up a serial console.

42 Unused GPIO pins The green GPIO pins are unused by default and so are the best pins to use in your hardware projects. The other pins may have more than one available purpose.

43 Soldering headers With the Pi Zero, you will need to solder GPIO headers onto the board. Using a reusable adhesive like Blu-Tack to hold the headers in place will make the job much easier for you.



44 Pin layout



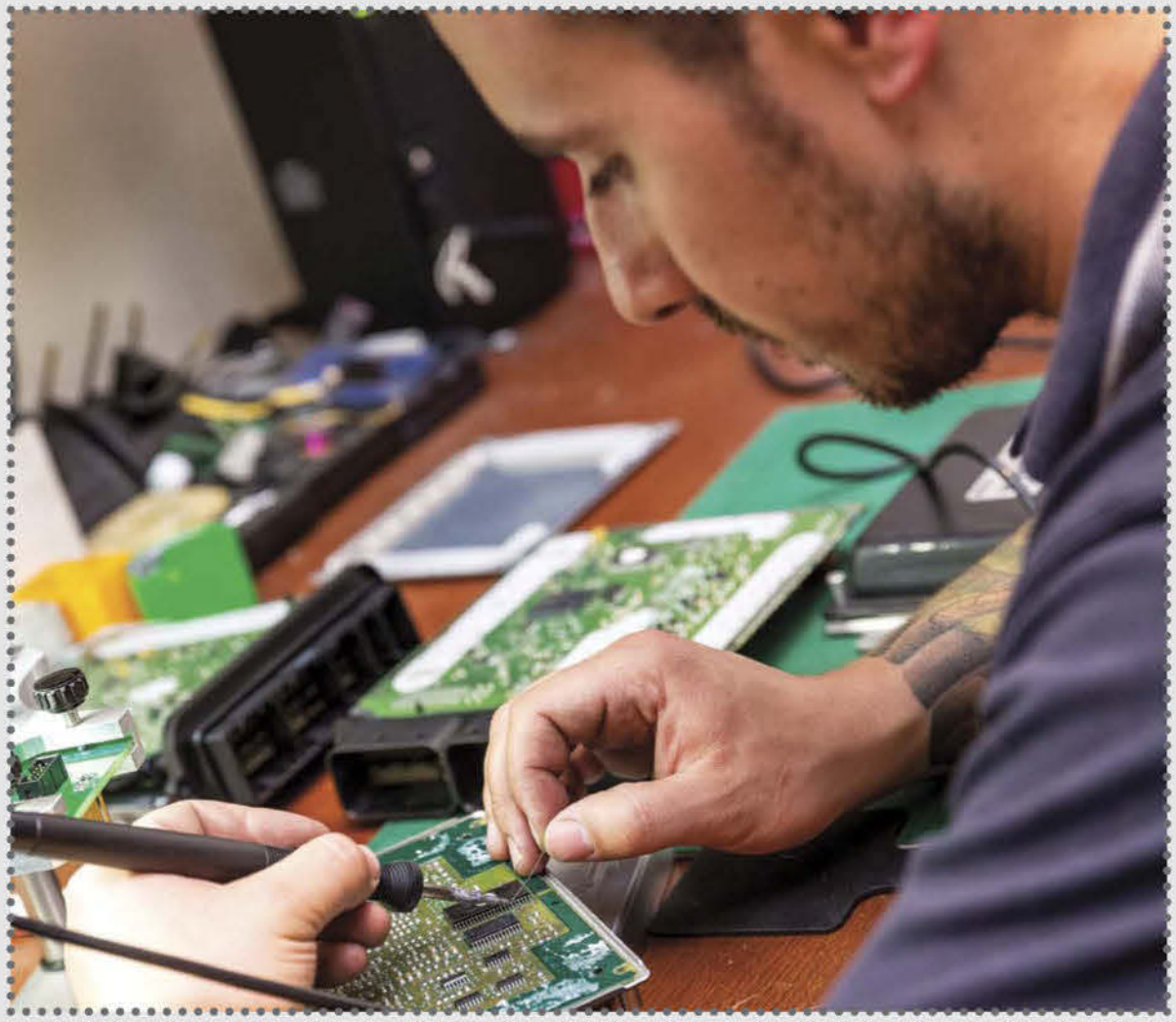
45 How to solder

Step One The tools

You need a soldering iron (30-40 watts, or ideally a temperature-controlled one), a stand to put it in, a damp sponge to clean the tip, and some thin solder. Lead solder is easier to work with than lead-free, and an iron with a square tip conducts heat better than a pointy one because of its larger surface area.

Step Two Soldering

Once the iron is hot, apply some solder to the tip of the iron and wipe off any excess on the sponge. When pressing the iron to the joint, the tip should be touching both the joint and the wire you are soldering. Do not apply solder directly to the iron; apply it to the joint/wire.



Step Three If you make a mistake

You probably won't need as much solder as you think you will, so be sparing. Still, accidents can happen, and the best way to remove excess solder is to get some de-soldering braid. This is copper that you put over solder, before putting the iron over the top. The solder will then be sucked up onto the braid.

Left Take your time when soldering and ensure your iron is kept in the stand when it's not in use

Five-minute practical fixes

46 Disable built-in sound card

If you are using a USB sound card then it can be easier to disable the built-in sound card completely:

```
sudo rm /etc/modprobe.d/alsa*
sudo editor /etc/modules
```

Change **snd-bcm2835** to **#snd-bcm2835** and save, then **sudo reboot**.

47 Assertions

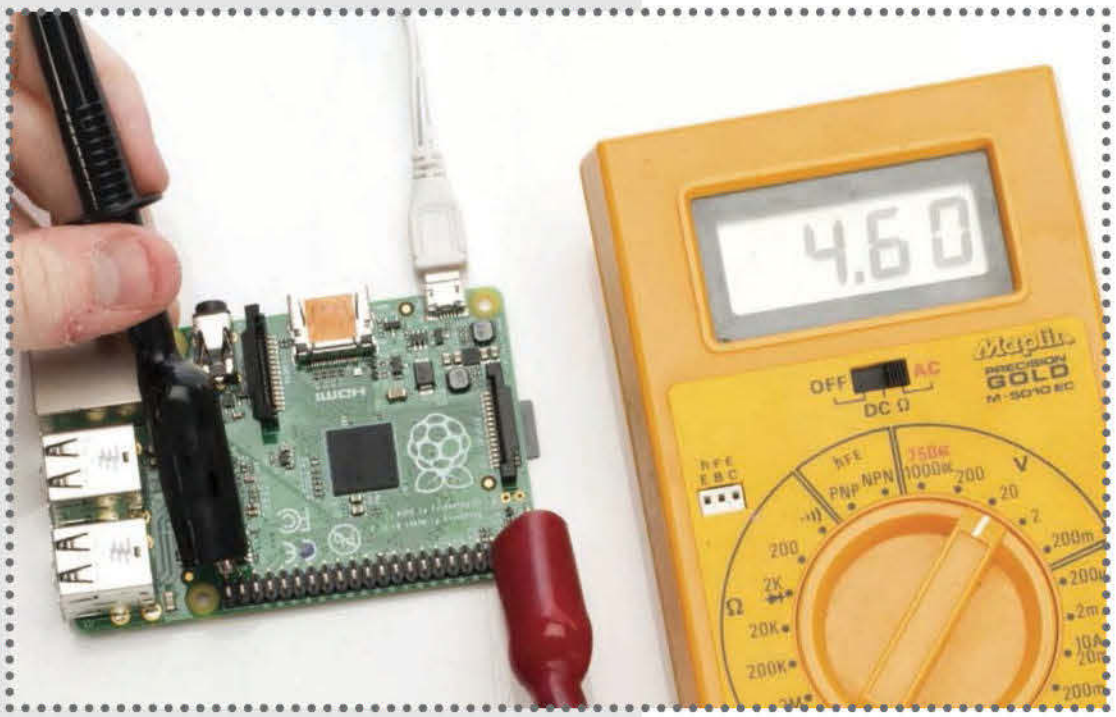
You can use **raspi-config** to set up a serial console so you can get a login shell using the UART0_TX and UART0_RX pins by connecting them to a USB to serial adapter, as explained in Tip 41:

```
sudo raspi-config
8 (Advanced Options)
A8 (Serial)
```

Select Yes to enable serial console. Finish, then reboot.

48 Use a multimeter to verify orientation

You can use a multimeter to verify the orientation of the GPIO pins. Looking vertically, with the USB ports at the bottom, the bottom-left pin (39) is ground and the top-right pin (2) is 5V. Using the negative terminal on ground and the positive terminal on 5V should show approximately 5V.



49 Need more current?

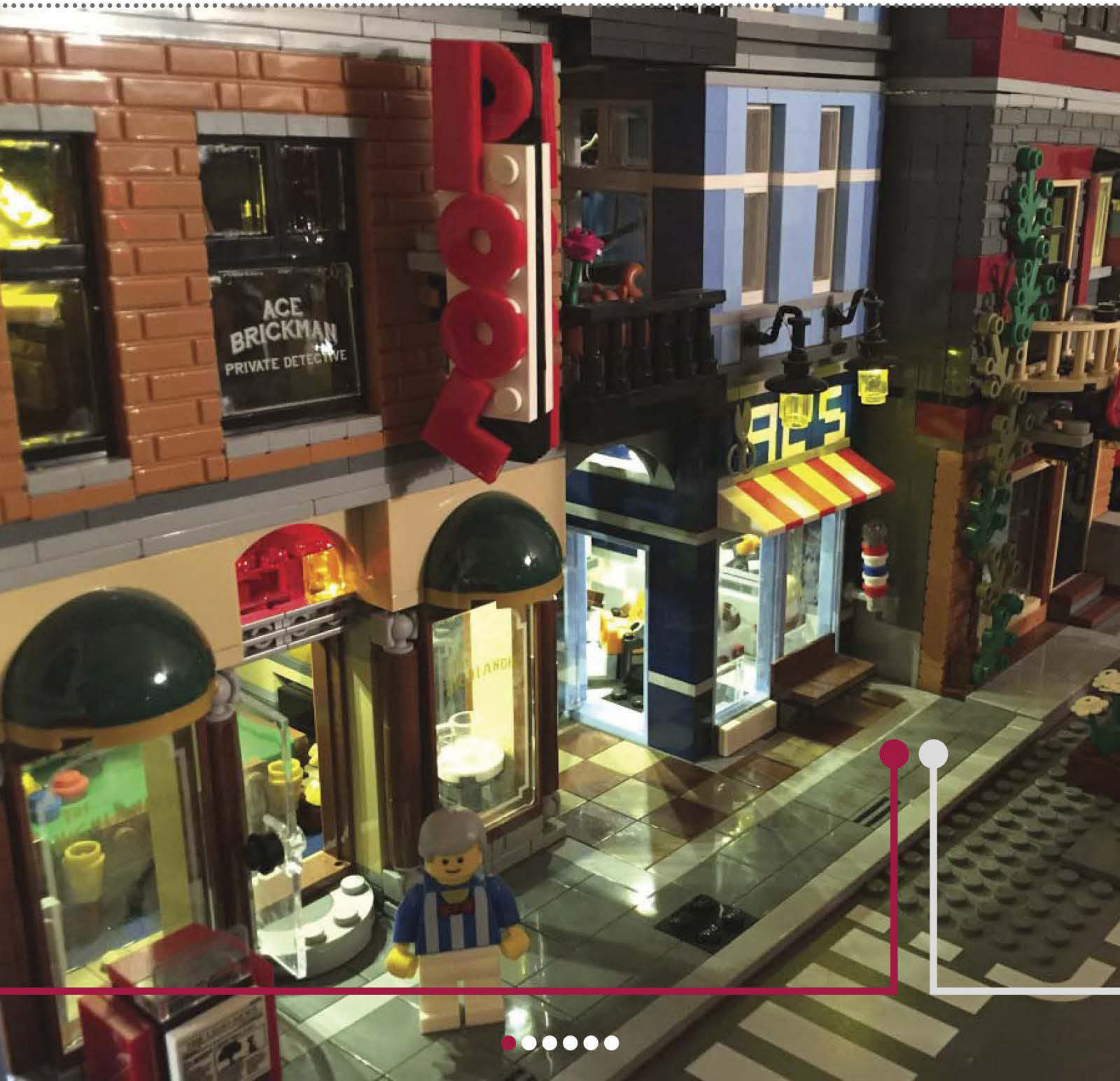
You should only draw a few milliamps of current from the GPIO pins. If you need more (or you need to switch a higher voltage), use the GPIO pin to switch a transistor connected to a stronger voltage source instead.

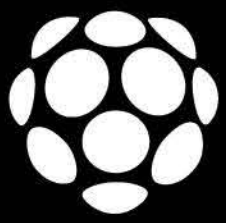
50 LED resistor calculations

To calculate an LED's resistance, use Ohm's law: Resistance = voltage / current. GPIO pin voltage is 3.3V. You need the voltage drop of the LED and its suggested current, so R = (3.3V – voltage_drop) / led_current. Calculate and round up to the next resistor.

Internet of LEGO

Cory Guynn brings an entire LEGO city to life using Internet of Things technologies



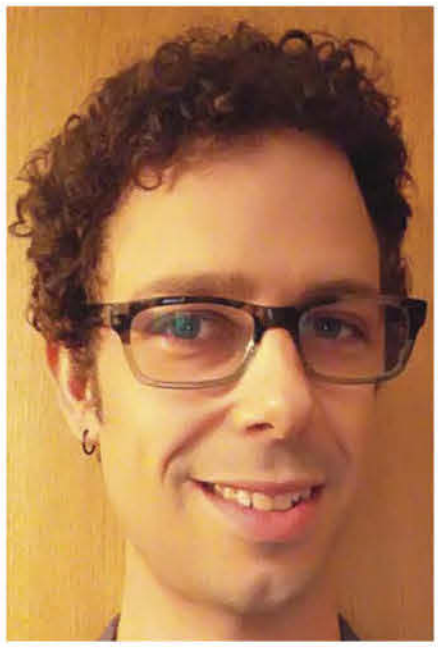


How long have you been working on this project?

I would say since September, when I really launched my blog. My wife got me a LEGO train set for Christmas last year and it got me back into LEGO. That's really how my engineering brain started as a kid – it was all about LEGOs. At the same time, I work for Cisco Meraki and we do a lot of stuff in the cloud – wireless and switching technologies – and the hot topic in Cisco is the Internet of Things, how it's going to be a multi-trillion dollar business over the next couple of years, and I wanted to wrap my head around it.

It's one of those elusive terms, like 'cloud'. What does 'cloud' really mean, right? So I started to look up all the technologies. I started with Node-RED, IBM's Node.js application for wiring things together, and that made it very easy to get the concepts; I'm not having to do a lot of coding at that stage. I started writing some front-ends to splash pages for Meraki, wireless hotspots… and once I understood what the concepts were, I started quickly picking up Node.js. And when I discovered Johnny-Five, which is this JavaScript framework for robotics, I was just hooked. It was very easy to understand.

I had a Raspberry Pi; I later bought an Arduino so I could do some servo controls. And then as I was running each of these various API sections, I was like, 'Okay, how do I control a servo? How do I control a light?' So I started making small programs and then combining that concept with LEGO, because I needed a medium to manipulate. I combined my hobby of LEGOs and my engineering brain with all these electronics and programming. It just made it a really fun and interesting project to work with. So that's where it started – I've opened Pandora's box now!



Cory Guynn is an IoT and cloud computing enthusiast, and a veteran Cisco Meraki systems engineer specialising in international deployments. To better understand IoT use cases, he has replicated them with LEGO and electronics.

How are you using things like Johnny-Five and Cylon.js?

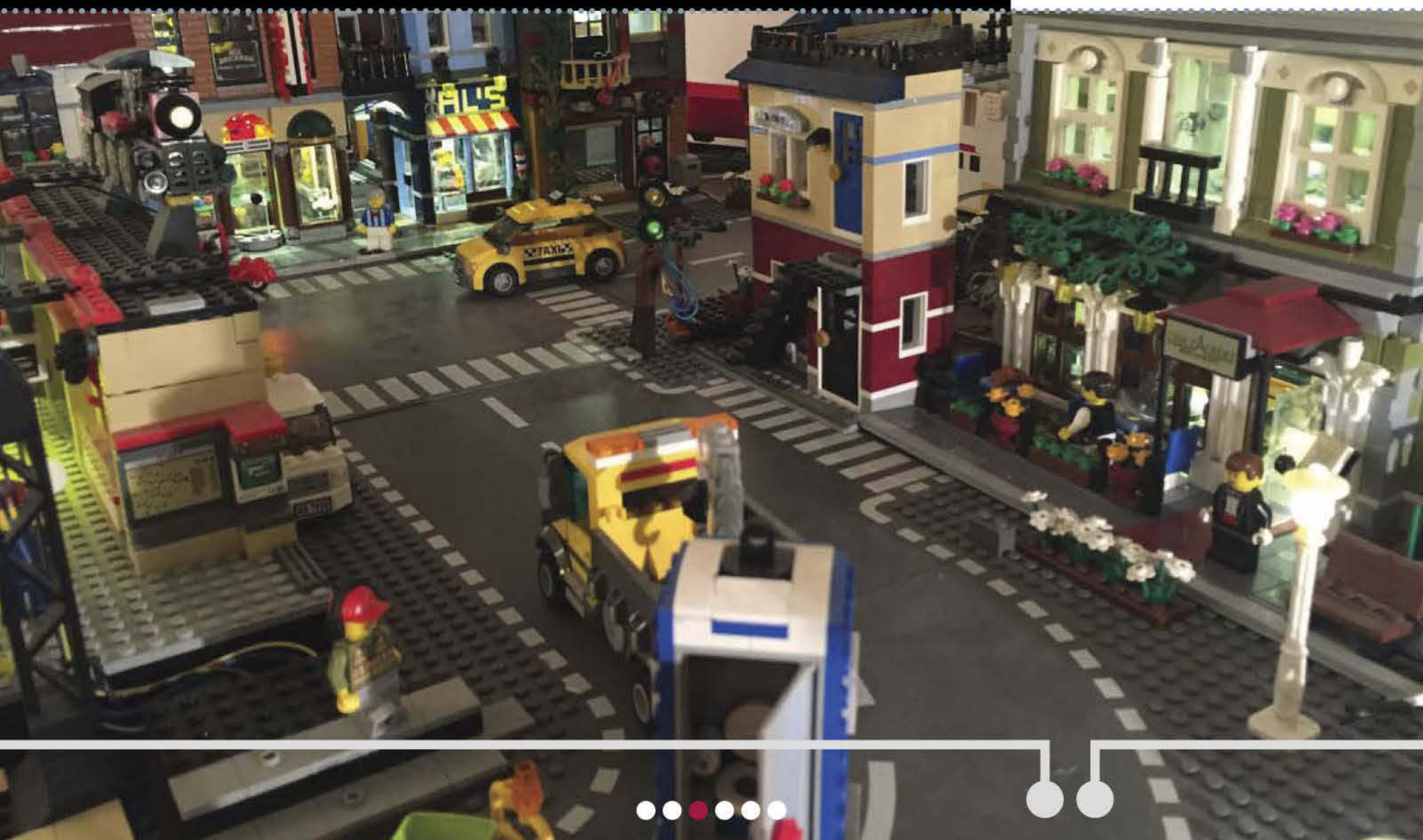
Johnny-Five and Cylon.js are more or less frameworks to quickly allow you to write code to control physical devices. And so, like Raspberry Pi traditionally uses Python or Arduino normally uses 'sketches', which are basically C++, Johnny-Five is JavaScript. I instantiate a servo by assigning it to a pin, and then just like writing a web page, if someone presses a button, it changes the state of that device. So it lends itself really well to robotics.

The difference is that if you use something like the normal Arduino sketches, it's very much loop-driven; you have a loop, it looks to a sensor, and if it's on it does something, and it just reiterates that loop. Whereas JavaScript is basically built around event-handling. So, I have a web page with tons of buttons, links, and it's just waiting for an activity – that's the change in paradigms.

So now I can say, 'If the sensor is triggered', like a button, 'then I do something'. With robotics, if I have a robot going

If you like

Cory is documenting the entire build process over at **internetoflego.com** and is also uploading the scripts for his smart city. From the city lighting and train automation through to the PubNub hub setup, it's all there for you to explore.



towards a wall, if the proximity sensor detects it's getting close then I change the servo on the wheels, and they change direction. That's why I like using JavaScript and Node.js as the server side. And Johnny-Five is really well-documented. They have an amazing book – it's Make: JavaScript Robotics – written by Rick Waldron.

Why did you decide to connect everything through PubNub?

I'm not partial to PubNub. I do like it, but it's one of those things where I was building an API for my city. My city is basically a Raspberry Pi and Arduino, and I wanted to now do more of the Internet of Things, whereas before it was just robotics, right? So I was trying to figure out a way to connect a website, other Wi-Fi-enabled devices, and I built a couple of versions of this: the initial one was a RESTful API. I would just go to some URL and it would turn on a light.

Then I decided to do web sockets in a traditional way and there were some limitations, because in that model the



Left A script on the Internet of LEGO blog publishes "WordPress visitor" to PubNub when a page is visited, which then fires up the disco lights

Further reading

SAM Labs launched an electronics kit earlier this summer that essentially combines LEGO and IoT: there are 15 individual blocks, each wireless and with a different sensor, slider or button, that can be snapped together: **samlabs.me**

Raspberry Pi had to be publicly accessible. Long story short, I was like 'Yes, I can do this', but I want to do something that's more of the publish-subscription model, where I can have random devices that can subscribe to a feed of data like a Twitter feed, and if it's of interest to them, they respond, and if they have an update, they blast it out to the other devices. There are numerous ways you could do this – socket cluster, things you can build yourself – but I found PubNub online and they had a really good explanation of how the concepts work. They had examples, they had SDKs that would work across Node.js, Arduino, Raspberry Pis.

Ultimately, it was a low barrier to entry and I was able to quickly build a server front-end to my city, add the little subscription code to it, and then I could easily add little hook-ins on my blog so if somebody goes to my blog, it would actually turn on lights in my city.

I'm working on an idea where I have Bluetooth Low-Energy beacons off my access point, and I can have an app on my phone so that if it detects the Bluetooth LE beacon, it publishes to PubNub and then that tells my city to start the train – maybe deliver a beer? [laughs] These are the sorts of things I have in my head!

What's next in line for construction?

I'm gonna build a weather station! I have temperature sensors, humidity sensors… I'm going to build a collection of those, and I've just now got this Arduino with Wi-Fi – the ESP8266 module – on it. Basically, I can now run an Arduino somewhere else, and I'm going to attach those sensors to it, then tie that back into the entire system and also combine it with a weather API, from weather.com, so I can see regional and local weather. Then I'm going to



Left Cory hacked his model train with LIRC and Node.js, using an infrared sensor to stop the train as it arrives at the station platform

have a scrolling screen within the city, maybe above the bank, saying 'The temperature is…' I'm just figuring out if I'm going to do it through PubNub or the more traditional model, which would be the MQTT standard for low-overhead communication per sensor.

I've done all of this – PubNub, web sockets, I'm looking at MQTT – and it's not that any one is always better; it's knowing when to use the right tools for the job. The better I understand all of the technologies available, this will help to figure out what's better for dashboards, better for sensors running off a battery, where energy and power is a concern… Now I can cherry-pick the right technology once I've really understood them all.

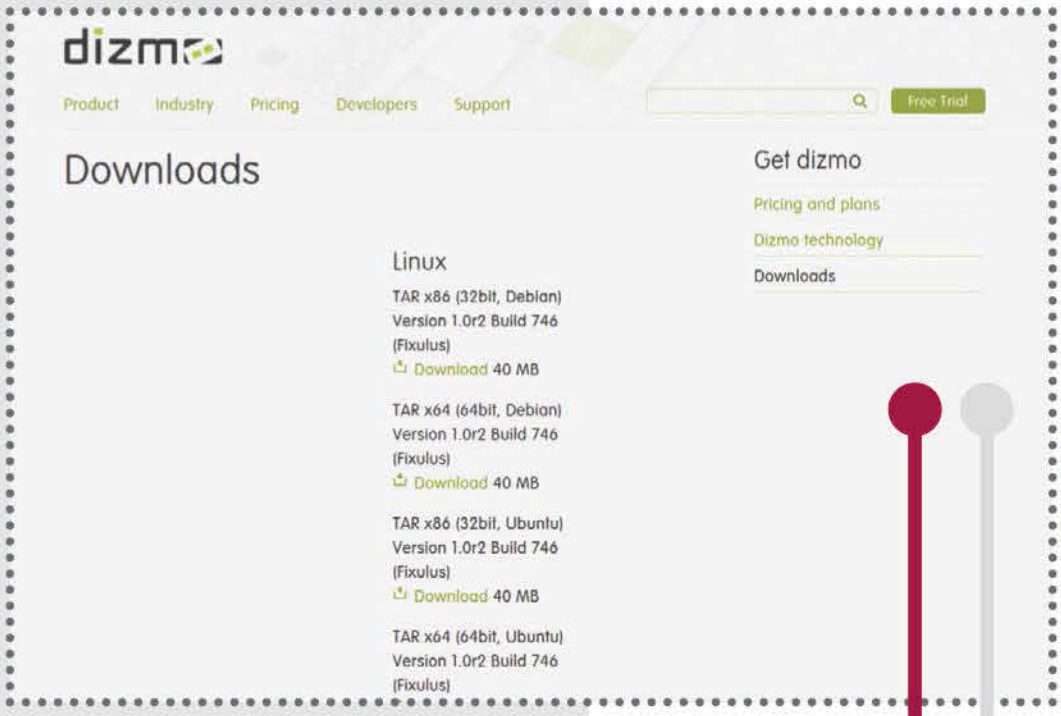
Ultimately, it's about creating concrete use cases in a medium that I can manipulate. As I talk about the Internet of Things to my customers at Cisco, thinking about the wider industry, it's one thing to talk about it, but once it's tangible, you can then easily scale that up to a smart city, a smart warehouse – you're just using real materials instead of little plastic bricks.

Monitor CPU temperature with Dizmo

Turn your Raspberry Pi into an Internet of Things hub with a widget-based interface

The Raspberry Pi is an exciting prospect for people interested in the Internet of Things – size, power and flexibility make it perfect for powering any Internet-connected device around the home or office. Setting up a Raspberry Pi to be the brain of an IoT network isn't exactly a case of selecting the right software in Raspbian, though; there's a lot of custom work you need to do to get one going.

This is where Dizmo comes in, enabling you to control IoT objects using an online API that you can then access remotely. To show you how it works, we're going to have it track the Raspberry Pi's core temperature. In this tutorial we are going to work entirely over SSH, but you can easily do this straight on the Pi – the benefit of SSH though is that it will be easier to maintain remotely.



Below Builds are available for various distros at **www.dizmo.com/** downloads. You can also find pricing plans



Above Dizmo is designed to be a multi-touch interface

01 Dial into your Pi

Make sure your Raspberry Pi can connect to your network, either via Wi-Fi or Ethernet cable, and find out the IP address by using **ifconfig**. Use this IP to dial into the Pi from another system with:

```
$ ssh pi@[IP address]
```

02 Install dizmoSpace

If you haven't already, head to **www.dizmo.com**, grab dizmoSpace and install it to the system you plan for it to work with. All you need to do is download the zip and unpack it, then click on the Dizmo icon in your dock or run it from the terminal.

03 Launch issues?

If Dizmo is complaining about libraries when you try to run it, you'll need to install some extra software. Open

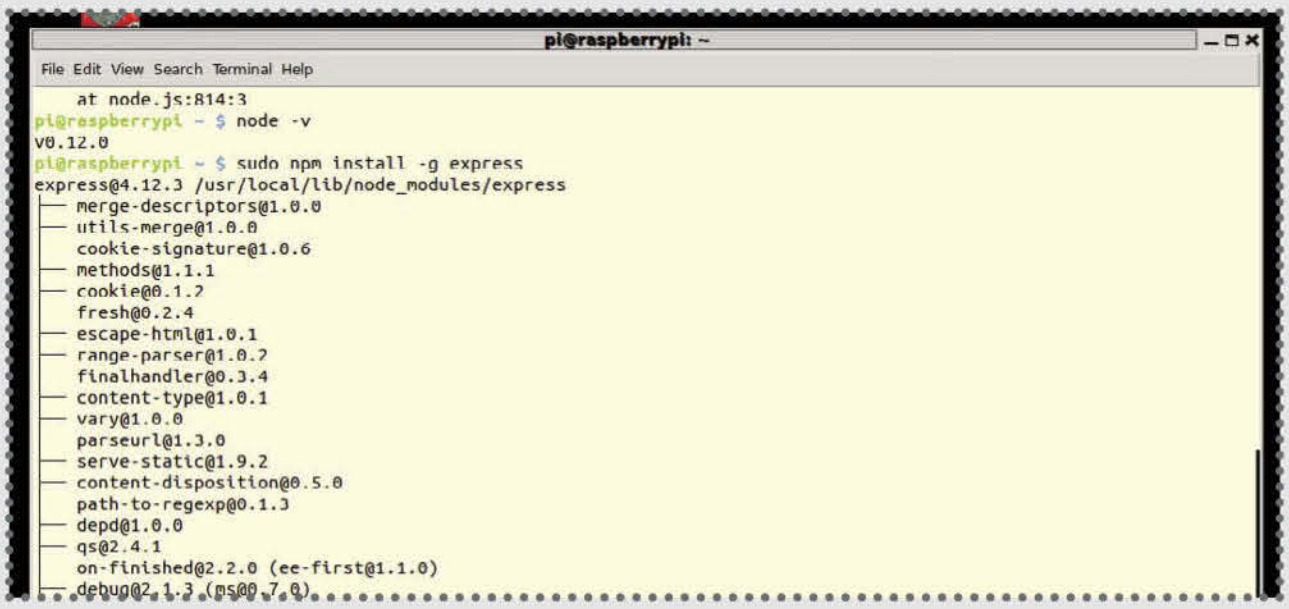
the terminal on the PC you're working from and install the extra software with the following:

```
$ sudo apt-get install libavahi-compat-libdnssd-dev
$ sudo apt-get install libavahi-client-dev
```

04 Download node.js

We need to grab the latest version of Node.js for the Pi. Back in the SSH connection to your Pi, use:

```
$ sudo wget http://node-arm.herokuapp.com/node_
    latest_armhf.deb
$ sudo dpkg -i node_latest_armhf.deb
```



05 Add framework

Use **node -v** to check if it's installed correctly – it should spit out a version number for you. Once that's done, install Express.js, which will be our web application framework:

```
$ sudo npm install -g express
$ sudo npm install -g express-generator
```

06 Install framework

We'll create the folder www in var and create a symlink in order for everything to run. Do this by moving to var, creating www and making the symlink with:

"Use node -v to check if it's installed correctly – it should spit out a version number for you"

```
$ cd /var
$ sudo mkdir www
$ cd www
$ sudo ln -s /usr/local/lib/node_modules
```

07 Package file

First, create the file package.json with sudo nano package.json, then enter:

```
{
“name”: “ServeSysinfo”,
“version”: “0.0.1”,
“dependencies”: {“express”: “4.x”}
}
```



08 App node

Now, create a file called app.js and enter the following:

```
var express = require(‘express’);
var app = express();
app.use(express.static(__dirname + ‘/public’));
app.listen(3000, function(){
console.log(‘listening on *.3000’);
});
```

“We’ll create the folder www in var and create a symlink in order for everything to run”

09 Start node.js

You can now start the Node.js server by typing in:

```
$ node app.js
```

It will say it's listening on *.3000. Start up a new terminal, **ssh** in, and create the folder /public with **mkdir /public** to save all of the CPU data in.

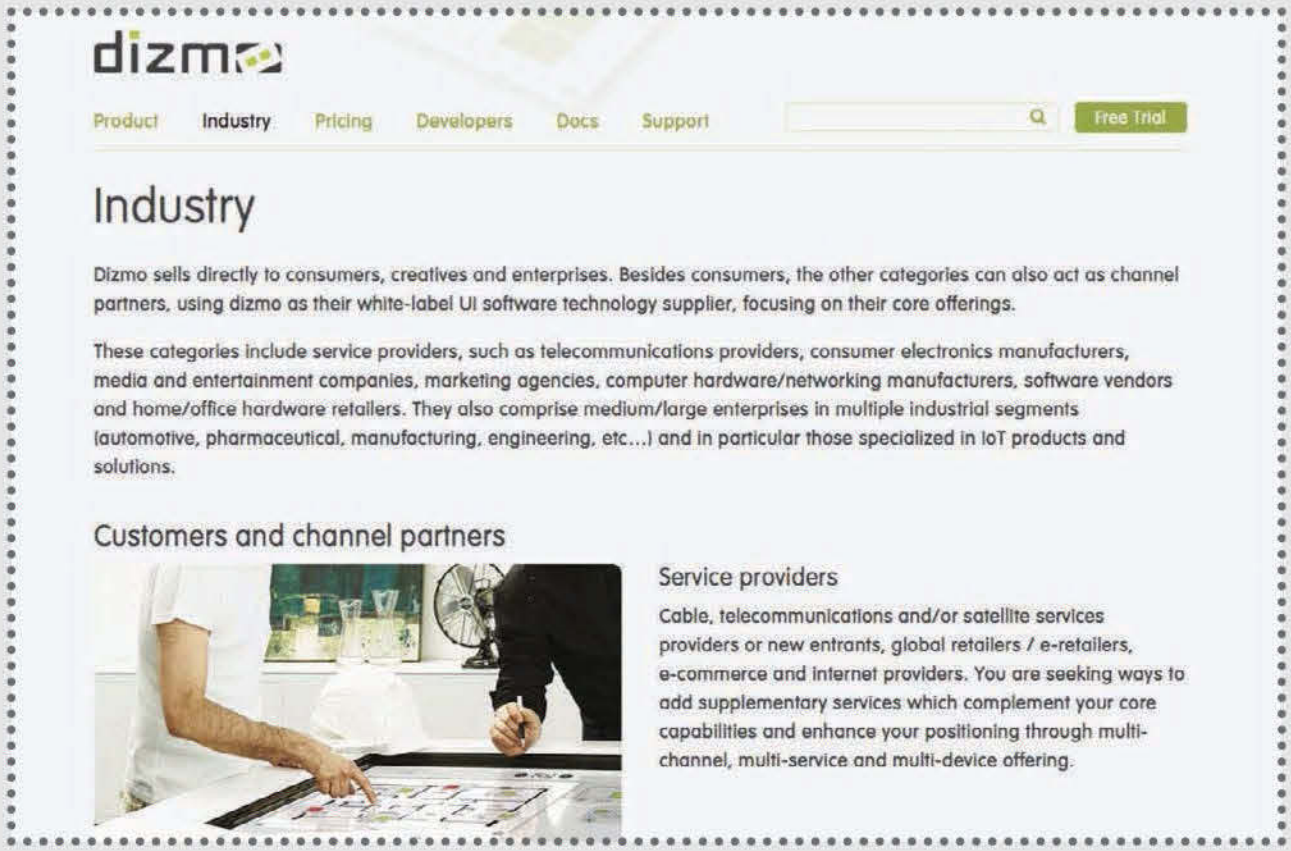
10 CPU information

We are going to use the **vcgencmd** command to get the CPU information from the Raspberry Pi. We will write a script that will do this and then write the info out to sysinfo.json. Download the file grabsysinfo.sh from **http://bit.ly/1P6Rx6X** and put it in /usr/local/bin.

11 Make a cronjob

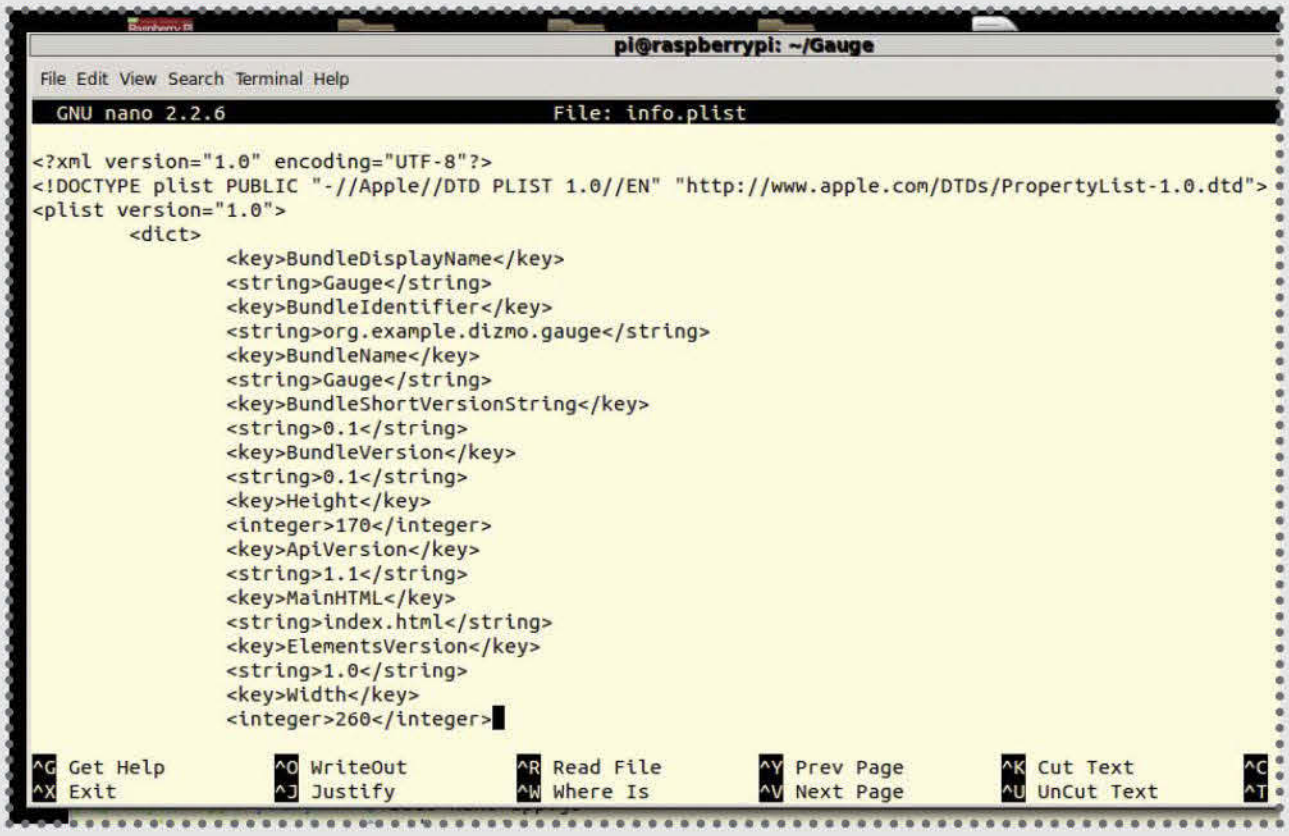
We will make it so that the temperature is updated every ten minutes. You can make it update much faster if you want, but have a play around with that. Open up cron with sudo crontab -e and add this at the end:

```
*/10 * * * * /usr/local/bin/grabsysinfo.sh
```



Left As it's multi-touch, Dizmo is perfect for interactive table displays in meetings

"We are going to use the vcgencmd command to get the CPU information. We will write a script that will do this and then write the info out to sysinfo.json"



12 Start creating the widget

It is time to actually start building the widget. First of all, create a folder on your local machine called Gauge and **cd** to it. Now you need to download the first file called info.plist into here by using the following:

```
$ wget x/info.plist
```

13 Index file

A Dizmo widget is basically a HTML file, packaging resources together to create an interface or graphic. Here, we have the main HTML file that uses jQuery, which helps display the temperature. Still in the Gauge folder, download it with:

```
$ wget x/index.html
```

14 Style guide

Now we'll add the CSS style sheet for the Dizmo widget. As usual, this styles up the display on the page that will become our widget. Download it with:

```
wget x/style.css
```

Internet of Things

It's not a very descriptive term, but the Internet of Things can be almost anything. Any item that is or can be connected to the internet or networks, such as modern automated lights, can be connected up to Dizmo and the Raspberry Pi.

15 Final application

The final step is to create the application.js file, which will call the temperature from the Raspberry Pi using Ajax. You can download it using:

```
wget x/application.js
```

Change the IP address to the one on your Pi.

Once that's done, you can test out the widget – compress the Gauge folder to a .zip and then change the .zip to a .dzm. Launch dizmoSpace and drag the dzm file onto it for it to start.

16 Get coding

With these building blocks, you can now start doing more interesting IoT things – controlling the GPIO ports, getting more information, having it connect to other objects to control them as well. Check out the Dizmo website for more details on projects that you can do.

Below We've gone for a simple CPU temperature gauge, but the possibilities really are endless

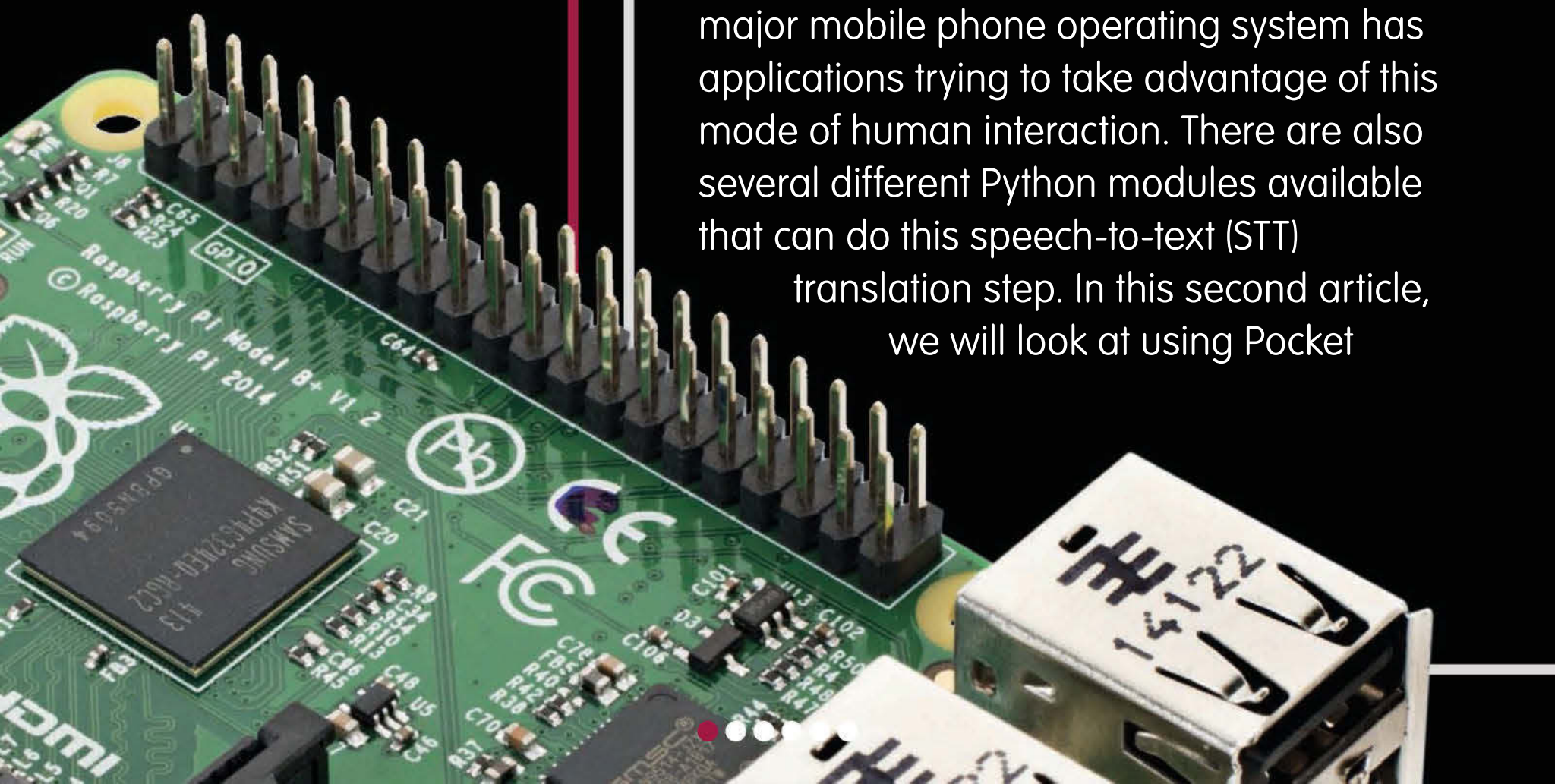


Create your own digital assistant – part two

In this instalment, teach J.A.R.V.I.S. to decode your audio and figure out which commands are being given

"There are several different Python modules available that can do this speech-to-text (STT) translation"

In the last issue we looked at how we could have our Raspberry Pis listen to the world around them. This is the first step in building our own version of the J.A.R.V.I.S system made famous in the *Iron Man* movies. The next step is to try and make sense of what we may have just heard. In general, this is called speech recognition and it is a very large and active area of research. Every major mobile phone operating system has applications trying to take advantage of this mode of human interaction. There are also several different Python modules available that can do this speech-to-text (STT) translation step. In this second article, we will look at using Pocket



Sphinx to do all the heavy lifting. Sphinx was developed by Carnegie Mellon University and is licensed under a BSD licence, so you are free to add any extra functionality that you may need for specific tasks. Because of the activity in this field, it is well worth your time to keep track of all the updates and performance improvements.

While you can download the source code for all of these modules and build it all from scratch, we are going to assume that you are using one of the Debian-based distributions, like Raspbian. For these you can simply use:

```
sudo apt-get install python-pocketsphinx
```

…to get all of the required files for the engine. You will also need audio model files and language model files in order to get a translation in your language of choice. To get the files needed for the English language, you can install the packages:

```
sudo apt-get install pocketsphinx-hmm-wsj1
    pocketsphinx-lm-wsj
```

You may need to go outside the regular package management system if you want to process other languages. Then you can simply start writing and using your code straight away. To start using these modules, you will need to import both pocketsphinx and sphinxbase with:

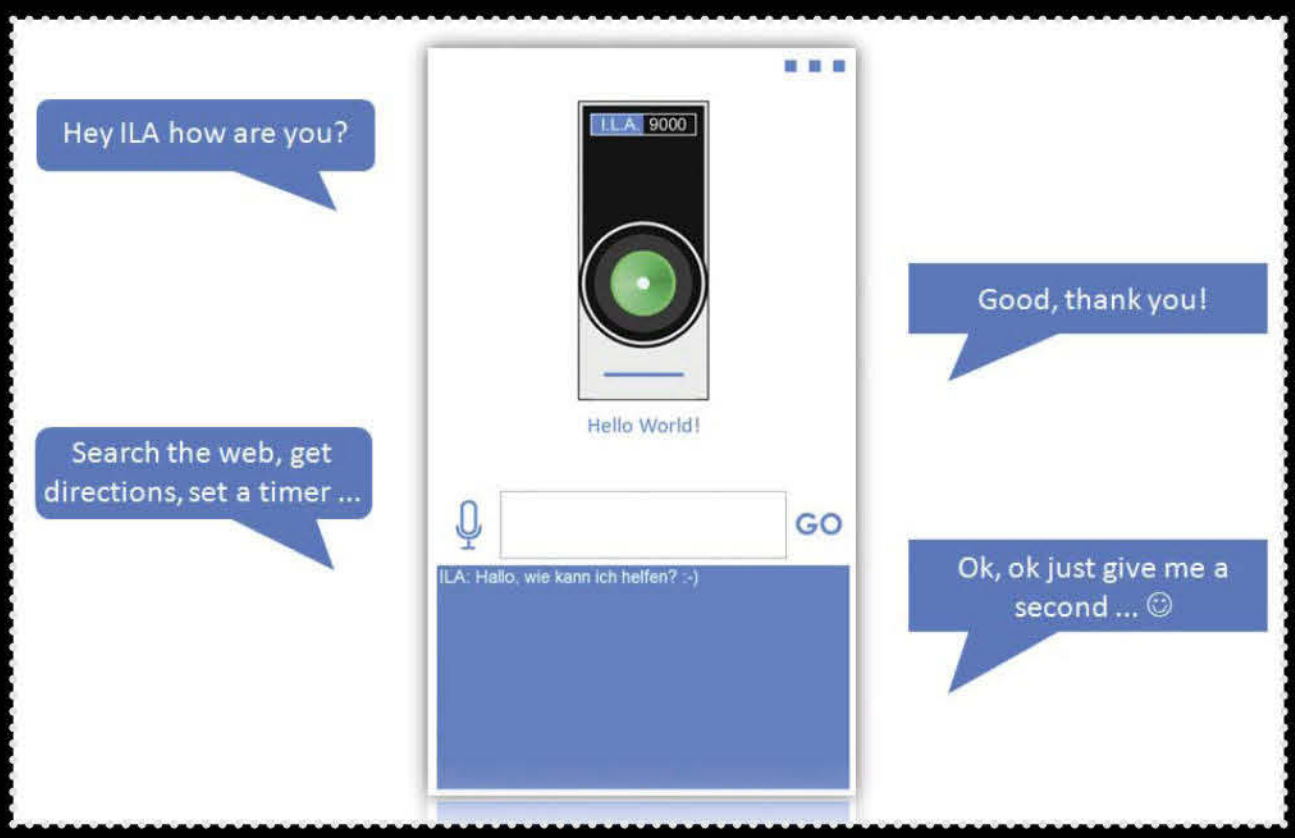
```
import pocketphinx as ps
import sphinxbase
```

These modules are actually Python wrappers around the C code that handles the actual computational

"Sphinx was developed by Carnegie Mellon University and is licensed under a BSD licence, so you are free to add any extra functionality that you may need for specific tasks"

work of translating sounds to text. The most basic workflow involves instantiating a Decoder object from the pocketsphinx module. The Decoder object takes several input parameters to define the language files it is allowed to use. These include 'hmm', 'lm' and 'dict'. If you use the above packages used to handle English, then the files you need will be in the directories /usr/share/pocketsphinx/model/hmm/wsj1 and /usr/share/pocketsphinx/model/lm/wsj. If you don't set these parameters, then it tries to use sensible defaults which usually work fine for English language speech. This newly created Decoder object can now be given WAV files with data to process. If you remember from last issue, we saved the recorded speech as a WAV file. In order to have this audio recorded in the correct format, you will want to edit the code from last issue and ensure that you are recording in mono (using one channel, for example), and recording at 16kHz with 16-bit quality. To read it properly you can use a file object and load it as a binary file with read permissions. WAV files have a small piece of header data at the beginning of the file that

"To have this audio recorded in the correct format, you will want to edit the code from last issue and ensure that you are recording in mono and recording at 16kHz with 16-bit quality"



Left CMUSphinx is used in cross-platform, open source projects like ILA, the Intelligent Learning Assistant

you need to jump over. This is done by using the seek function to jump over the first 44 bytes. Now that the file pointer is in the correct position, you can hand the file object in to the Decoder object's decode_raw() function. It will then go off and do a bunch of data crunching to try and figure what was said. To get the results, you would use the get_hyp() function call. You get a list with three elements from this function: a string containing the best guess at the inputted spoken text, a string containing the utterance ID and also a number containing the score for this guess.

So far, we've looked at how to use the generic language and audio models for a particular language. But Pocket Sphinx is a research-level language system, so it has tools available to enable you to build your own models. In this way, you can train your code to understand your particular voice with all of its peculiarities and accents. This is a long process, so most people will not be interested in doing something so intensive. However, if you are interested in giving this a try, there is information available at the main website (**http://cmusphinx.sourceforge.net**). You can also define your own models and grammars to tell pocketsphinx how to interpret the audio that it is processing. Once again, effectively carrying out these tasks will require more in depth reading on your part.

If you want to process audio more directly, you can tell Pocket Sphinx to start processing with the function start_utt(). You can then start reading audio from your microphone. You will want to read in appropriate sized blocks of data before handing it in to pocketsphinx – specifically to the function process_raw() – and you will still need to use the function get_hyp() to actually get the translated text. Also, because your code can't know

"Pocket Sphinx is a research-level language system, so it has tools available to enable you to build your own models. In this way, you can train your code to understand your particular voice with all of its peculiarities and accents"

when someone has finished a complete utterance, you will need to do this from within a loop. On each pass of the loop, read another chunk of audio and feed it into pocketsphinx. You then need to call get_hyp() again to see if you can get anything intelligible from the data. When you are done doing this real-time processing, you can use the function end_utt().

So far, we have covered how to record your speech and how to turn that speech into text. In the final part of this tutorial next issue, you will learn how to take that translated speech and actually take actions based on how the system has been configured. But even with only these two steps, you could build yourself a nifty little dictaphone or vocal note-taking system.

> "On each pass of the loop, read another chunk of audio and feed it into pocketsphinx. You then need to call get_hyp() again"

The Code DIGITAL ASSISTANT

```
# You first need to import the required modules
import pocketsphinx as ps
import sphinxbase

# Next, you need to create a Decoder object
hmmd = '/usr/share/pocketsphinx/model/hmm/wsj1'
lmd = '/usr/share/pocketsphinx/lm/wsj/wlist5o.3e-7.vp.tg.lm.DMP'
dictd = '/usr/share/pocketsphinx/lm/wsj/wlist5o.dic'
d = ps.Decoder(hmm=hmmd, lm=lmd, dict=dictd)

# You need to jump over the header information in your WAV file
wavFile = file('my_file.wav', 'rb')
wavFile.seek(44)

# Now you can decode the audio
d.decode_raw(wavFile)
results = d.get_hyp()
```

The Code DIGITAL ASSISTANT

```
# The most likely guess is the first one
decoded_speech = results[0]
print “I said ”, decoded_speech[0], “ with a confidence of ”, decoded_speech[1]

# To do live decoding, you need the PyAudio module
import pyaudio
p = pyaudio.PyAudio()

# You can now open an input stream
in_stream = p.open(format=pyaudio.paInt16, channels=1, rate=16000,
        input=True, frames_per_buffer=1024)
in_stream.start_stream()

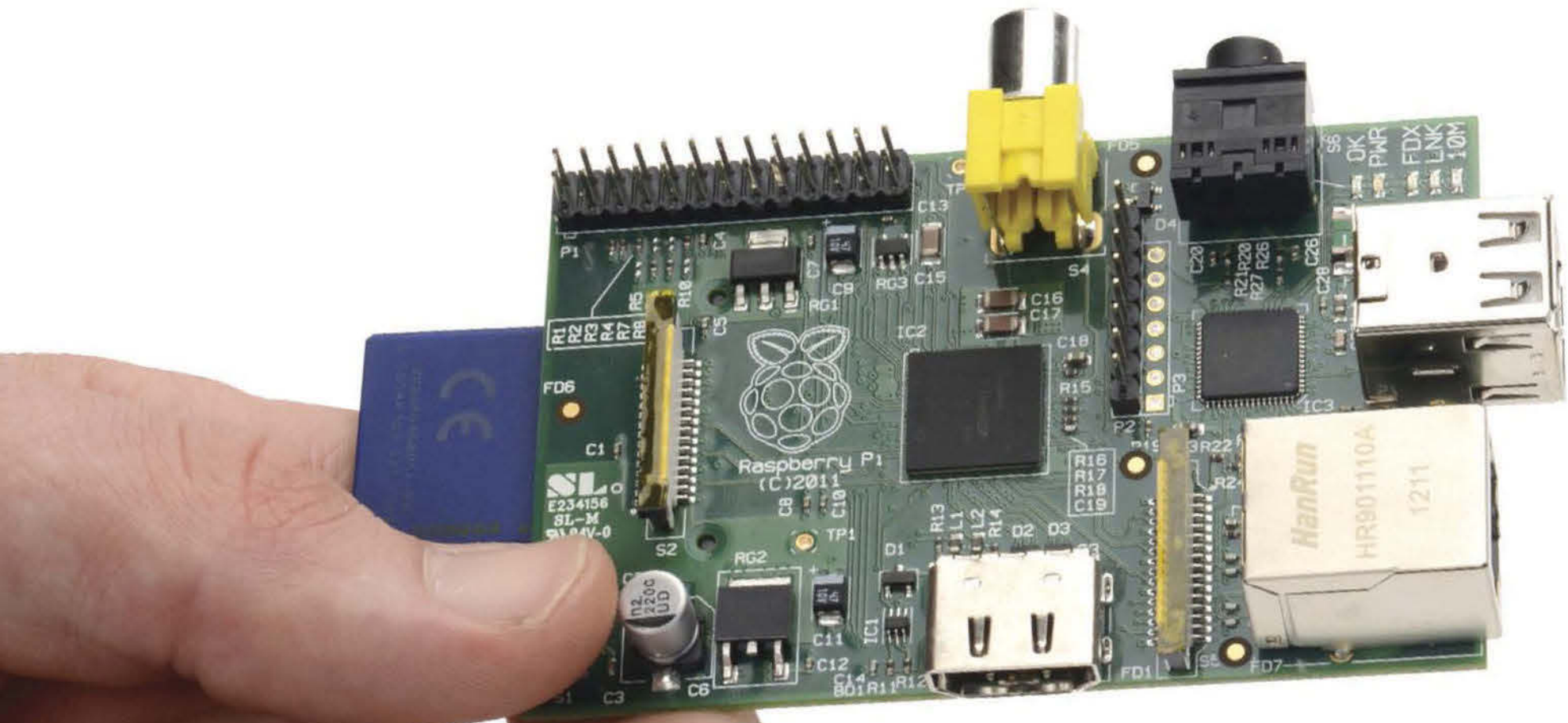
# Now you can start decoding
d.start_utt()
while True:
   buf = in_stream.read(1024)
   d.process_raw(buf, False, False)
   results = d.get_hyp()

   # Here you would do something based on the decoded speech
   # When you are done, you can shut everything down
   break
d.end_utt()
```

Talking Pi

Join the conversation at...

@linuxusermag | Linux User & Developer | @ RasPi@imagine-publishing.co.uk



The Raspberry Pi encourages a lot of hands-on work and this means it doesn't quite work like – or as easy as – your laser focus-tested smartphone interface, or even a normal computer. So we're answering your burning Raspberry Pi questions each issue – get in touch with us on Twitter, Facebook or by email.

How can I set up my Pi Zero so I can access it remotely, but without using a powered USB hub?

Cait via email

If you don't have a powered USB hub then you can't connect your keyboard and mouse at the same time as a Wi-Fi dongle, and that's the easiest way to get configure the connection. But not to worry – you can instead use another computer to tweak the network interfaces file on your SD card, then plug it back into the Pi once you're done (see this guide: **bit.ly/1LNCWIW**). Then, you can have just have the dongle in the mini-USB data port and SSH in.

Keep up with the latest Raspberry Pi news by following @LinuxUserMag on Twitter. Search for the hashtag #RasPiMag

JUST A SCORE

WHAT'S YOUR JUST A SCORE?

Have you heard of Just A Score? It's a new, completely free app that gives you all the latest review scores. You can score anything in the world, like and share scores, follow scorers for your favourite topics and much more. And it's really good fun!

Do you think I could fit a Pi Zero inside a NES controller?

Ryan via Facebook

Absolutely – someone's already beaten you to it! Check out their write-up on Hackaday: **hackaday.io/post/28804**. Basically, you need to crack open the controller and find a spot to mount the Zero, then cut out holes in the side for its ports. Next, you need to wire up the controller's PCB to your Pi's GPIO pins. This bit changes from controller to controller, so you'll need to read around if you go for something other than a NES controller. With that done, you re-mount the buttons, close the case up and then get started on your RetroPie setup!

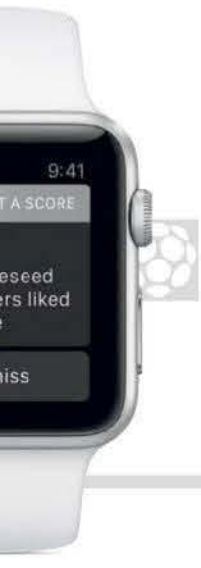


I can never remember which GPIO pin is which. Do you have a cheat sheet?
Si via Twitter

If you swipe back to this issue's Pi Zero feature then you'll find a handy pinout illustration that should help – it identifies both the physical pin numbers and also the Broadcom pin numbers. But even handier would be one of the RasPiO port labels boards – they cost just a couple of pounds and are small boards with holes cut out for the GPIO pins, available for both the 26- and 40-pin varieties. You slide one over the top of your Pi's GPIO header and then just read off the labels that neatly line up with each pin.

I need to have dozens of screens displaying the same animation. What's the simplest way?
Karen via email

Screenly (**screenlyapp.com**) is probably your best bet – there are ways to set this up manually, but they can be a pain! Screenly is a digital signage software that comes in Pro and Open Source Edition variants. Pro is simpler to use but you need to subscribe, and more screens costs more money. Screenly OSE can do the same job for free but you'll need to set up your nodes manually, though it's significantly easier than coding a custom solution. Plus, you just need the one Pi to manage all your nodes.



JUST A SCORE

WHAT'S YOUR JUST A SCORE?

You can score absolutely anything on Just A Score. We love to keep an eye on free/libre software to see what you think is worth downloading...

10 LinuxUserMag scored **10** for Keybase

9 LinuxUserMag scored **9** for Cinnamon Desktop

8 LinuxUserMag scored **8** for Tomahawk

4 LinuxUserMag scored **4** for Anaconda installer

3 LinuxUserMag scored **3** for FOSS That Hasn't Been Maintained In Years

SCORE ANYTHING

JUST A SCORE

Download on the App Store

Next issue

Get inspired **Expert advice** **Easy-to-follow guides**

20 AWESOME PROJECTS

Made by you



Get this issue's source code at:

www.linuxuser.co.uk/raspicode